1-2-2013

The Drosophila Interactions Database: Integrating The Interactome And Transcriptome

Thilakam Murali
Wayne State University,

THE DROSOPHILA INTERACTIONS DATABASE: INTEGRATING THE INTERACTOME AND TRANSCRIPTOME

by

THILAKAM MURALI

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2013

MAJOR: MOLECULAR BIOLOGY AND GENETICS

Approved By:

_____

Advisor Date

_____

_____

_____

DEDICATION

To my mother and my daughter

To my guru

ACKNOWLEDGEMENTS

Over the past six years I have received support and encouragement from many people. First of all, I would like to thank my mentor Russell L. Finley Jr., for an incredible, thoughtful and rewarding journey of learning and discovery. He has fostered my love of science and has encouraged me to ask difficult questions. His insistence on clarity in speaking and writing has made me a better communicator. Last but not the least, without him hiring me as research assistant in 2007, I would not be writing this thesis and probably would not have ventured into an academic department again.

I would like to thank Dr. Gerardus Tromp and Dr. Helena Kuivaniemi for giving me a 'home' in Wayne State. I really had no place to go hanging in between the Biology department and nowhere. Not only did they give me a place to work at but also helped me with my application to CMMG and taught me human genetics. My special thanks to Dr. Tromp for teaching me awesome PERL and R, advising me on statistics and serving on my committee.

I would like to thank my committee member Dr. Derek Wildman for his advice and insightful comments and also his advice in my early days of transitioning into CMMG. I would like to thank Dr. Leonard Lipovich for serving on my committee and for his advice and thought-provoking comments.

I would like to thank my friends Seeta Nyayapathy and Wes Colangelo in Biology without whose encouragement and support I would not have moved to CMMG. I would like to thank Nermin for all her help, proof-reading my thesis and for valuable feedback

and for being my friend. I would like to thank Julie Hines who made me laugh when times were difficult and she still does whenever I get to see her. My thanks to Paul Albosta for his comments and feedback on my work. Many thanks also to all the former members of the Finley Lab, especially Jingkai Yu whose project I inherited, Lana Pacifico for being a coauthor and Steve Guest for thoughtful comments and help with my first publication. I would also like to thank Dumrong for letting me work on his project, for feedback about mine; I enjoyed the long discussions about international politics and debated if PERL is better than Python - of course PERL is better Dumrong!

I would like to thank my loving family for their warm hugs, believing in me, and encouraging me throughout. The family wisdom and the wisdom of the elders were constant reminders for me to apply myself systematically in spite of many hurdles and roadblocks. I was often told "Enjoy the ride as destinations just come and go!", "Aim high", "Enjoy what you do and do not worry about the results."

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Biological Systems

The basic unit of a living system has been recognized as a cell [1]. Multicellular systems have coordinated groups of cells working together to form tissues and organs. Unicellular systems such as bacteria and yeast share common features with cells of multicellular systems; studies on unicellular systems have helped us understand common pathways that have been conserved from yeast to higher organisms including human [2]. The characteristic features of biological systems are the capacity to develop, maintain and adapt themselves to the environment and reproduce. The study of a biological system is the study of the parts encoded by its genome and how the parts come together to perform the functions of the system.

There are more than 20,000 protein-coding genes and an equal number of genes coding for RNA products in the human genome with more likely to be discovered [3], while the genome of the model organism *Drosophila* has more than 15,000 protein coding genes and hundreds of RNA encoding genes [4]. The central goal of most biological research is to discover how genes function in specific systems. Currently we do not know the functions of the majority of them. In general, the protein and RNA molecules catalyze biochemical reactions, transfer signals from the extracellular environment to the nucleus, organize cellular compartments and regulate cellular physiology and structure. The genome is transcribed by the action of other proteins and

RNA during development and in response to the environment [5, 6]. It is also maintained and faithfully copied during cell division.

Biological systems are rarely static entities. They grow, differentiate, respond to the environment and also maintain homeostasis. There are different types of cells in a multicellular system which form organs and tissues; these cells have varying complements of proteins and RNA while having the same genome [7]. This change in the protein and RNA complement is brought about by changes in the expression of genes in response to developmental and environmental cues which in turn lead to changes in structure and function, namely phenotypes [5]. The physical interaction of various molecules such as DNA, RNA and proteins with each other form complex assemblies which perform different biological functions. A particular biological function or phenotype can therefore be assigned to the entire set of interacting parts or network of interacting molecules and rarely to any particular molecule [2, 8, 9]. It has become apparent in studies of the different molecules in biological systems, that the various functions of a cell are performed by different groups of interacting molecules that are distinguishable from each other. As a result, a biological system has been viewed as organized modularly [2, 8, 10].

Development of high throughput technologies in molecular biology have made it possible to build several types of molecular networks enabling studies to move from the single molecule level to the systems level [11-13]. The molecular networks of the cell such as protein-DNA interaction networks and protein-protein interaction networks have

been studied to get functional details of systems [14-18]. Apart from studies on molecular networks, studies at the systems level also include studies on transcriptomics, proteomics, phenomics, metabalomics, genetic interactions, human disease gene studies and many others [19].

While it is important to study the different types of data by themselves, in order to really gain a deeper understanding of biological systems it is necessary to integrate the various types of data, as any system function is dependent on how all of its molecular parts come together [9]. In order to use the deluge of genome-wide data, formalisms from computer science and mathematics have been borrowed - this has led to a renewed interest in the field of systems biology after a hiatus into reductionist approaches to studying living systems [9, 20-25]. It has also been proposed that the study of systems biology should also be a 'search for organizing principles' as biological systems are by far the most complex systems known and building predictive models may be difficult in the near future [15, 26]. The central Aristotelian idea behind systems thinking is that the properties of the biological system 'emerge' as a result of the interactions of all of its molecular parts and therefore it is necessary to study the system as a whole [9].

1.2 Summary

In section 1.1, I discussed biological systems, the most complex systems known and the need to study them at the systems level including all the parts of the system and their interactomes. In section 1.3, I discuss the different types of interaction networks and how they have been used to answer biological questions. I also discuss the shortcomings

of the network data that have been collected thus far, including the fact that they are only composites of potential interactions that can take place in an organism. The composite nature of these networks is mainly due to the fact that the techniques used to identify the interactions do not take into account the *in vivo* spatiotemporal context or the differential expression of genes in different contexts. In section 1.4, I discuss the importance of genome-wide, organism-wide and tissue-wide gene expression analysis and how it might be used to provide context to the interaction data thereby making the data more biologically useful. Finally in section 1.5, I explain the need for the integration of various types of genome-wide data and various types of resources for these data. I introduce the *Drosophila* interactions database, DroID, which allows the integration of heterogeneous data to enable better understanding of biological systems.

1.3 RNA, DNA and protein interaction networks

Cells are complex assemblies of interacting molecules including genes and gene products. Mapping the interactions between these molecules is an important step in identifying the functions and pathways they are involved in [27]. These interacting molecules may be represented as a network or graph with edges connecting the molecules that interact [17]. An edge can be undirected where it simply denotes that two molecules bind to each other or directed where one molecule binds the other and modifies or regulates it. Representing interactions as a graph or network enables the application of graph theory and computational methods to extract biological information from these networks [17]. Interactome network data can be used to understand individual biological

processes and the organization of entire biological systems. In this section I am going discuss protein-protein and protein-DNA interaction networks, which are physical interaction networks, and RNA-gene networks. I will also discuss the types of biological information gained from network studies.

1.3.1 Protein-Protein Interaction (PPI) network

Protein-protein interactions are fundamental to the structure and function of biological systems. In order to understand how proteins come together to form macromolecular structures and pathways, large scale PPI mapping studies have been undertaken for yeast [28-32], worm [33], *Drosophila* [34-38] and human [39-41]. The majority of available PPI have come from two methods, yeast two hybrid [42] and protein complex determination [43]. In the yeast two hybrid system interactions between pairs of proteins expressed in a yeast nucleus are detected using reporter gene assays. In complex pull downs, a tagged protein along with all the proteins that interact with it are purified from cells, the composition of the complex is determined using mass spectrometry, and interactions between members are inferred. Most of these studies involve forced expression of the tagged bait proteins in cultured cells. Recently, complexes in cell culture systems have been mapped using cell fractionation followed by mass spectrometry bypassing the step of tagging proteins [44]. The yeast two hybrid system and co-affinity purification followed by mass spectrometry are two complementary methods that have been amenable to scale up or high throughput detection of PPI. The yeast two hybrid system system detects binary and also transient

interactions between proteins in contrast to the affinity purification followed by mass spectrometry method which is more suited to detecting stable molecular machines or complexes [45]. Both of these methods are biased towards intracellular and nuclear interactions. Recently, yeast membrane complexes were affinity purified and subjected to MS and a membrane protein interaction map was assembled [29]. Such studies could potentially reduce the bias against membrane protein interactions in available PPI data.

It is important to note about all these methods that the interactions are detected under artificial conditions and many may not be physiologically relevant. Also the interactions detected are snap shots of potential interactions in different spatial and temporal conditions. In order to identify the interactions in different cell types or under different dynamic conditions, interactions have to be mapped *in vivo* under the relevant conditions or they have to be integrated with relevant *in vivo* data such as gene expression data.

The PPI detected by various high throughput methods along with manually curated low throughput interactions are stored in large databases such as BioGRID, IntAct and MINT, to name the major databases [46-48]. This allows users to query the data for studies of individual proteins or for systems biologists to download entire data sets including hundreds of thousands of interactions for large scale systems level studies.

1.3.2 Protein-DNA Interaction (PDI) network

Protein-DNA interactions (PDI) or more specifically transcription factor (TF)-gene interactions are those that take place between a TF and a specific DNA sequence of

a gene to regulate its expression. PDI networks are therefore directed networks. These networks play major roles in development and also regulate responses of cells to the environment [49]. TFs bind combinatorially to the upstream regions of genes [50]. This combinatorial regulation ensures robust developmental and environmental responses, helps buffer responses in spite of fluctuations in levels of gene products, and helps to integrate signaling inputs of various strengths and durations [49]. The goals of PDI mapping studies have been to understand gene expression regulation and predict the outputs of different combinatorial regulation events by constructing models using changing patterns of gene expression that lead to developmental events [49]. TFs regulate each other by participating in protein-protein interactions and also by binding to and regulating the expression of another TF-encoding gene [49, 50].

TF-gene interactions are being mapped on a genome-wide scale by methods such as chromatin immuno-precipitation (ChIP) or DNA protection assays [3, 51, 52]. The sequence to which a TF binds along with the expression pattern of the target is often used to infer a regulatory interaction between a TF and its target gene and to build PDI networks [51-54]. In ChIP assays, a tagged TF is expressed in cultured cells and the TF-bound DNA is isolated and the sequence is identified and mapped to the genome. The resulting interactome networks are thereby composites of hundreds of thousands of potential interactions that can take place in an organism. TF-gene regulatory network maps have been built with these data enabling the analyses of these networks to identify biologically important regulatory motifs [54].

Finally in order to find out if an interaction is functional it needs to be integrated with gene expression data (detailed in the next section) along with other data such as chromatin marks that indicate activation or repression of genes [3, 51, 55]. DNase I footprinting [56] also appears to be a promising way to get spatial or temporal context-specific TF binding sites in the genome [3, 54].

1.3.3 RNA-gene network

Gene expression can also be controlled by regulatory RNAs or non-coding RNAs (ncRNAs). The ncRNAs that I am concerned with here are those that regulate specific genes. Thus, I will refer to these functional interactions together as "RNA-Gene" interactions. microRNAs (miRs), for example, post-transcriptionally regulate specific genes by base pairing with target mRNAs. The primary strategy to identify miRNAs and their target genes has been computational prediction based on sequence complementarily to known protein coding genes. Only a small fraction has been experimentally verified thus far [57]. Currently, well-established algorithms are available for prediction of small ncRNAs such as miRs and their targets [58, 59]. The predicted RNA-gene interactions are stored in many databases, examples being TargetScanS [60] and MinoTar [61].

1.3.4 Network Studies

Both PPI and PDI networks are large networks with hundreds of thousands of interactions. Network representation of PPI and PDI have enabled scientists to use mathematical methods, specifically graph theory to study these networks [17]. Network motifs, modules (communities or sub-graphs), and hierarchy have been studied using

graph theory [62]. Disease genes have been prioritized for further study using network algorithms that identify modules that include known disease genes. For the majority of human diseases we do not know how specific genes contribute to the disease. It has been shown that often a human disease is caused by mutations in different genes in the same pathway or closely related pathways (for examples see [63, 64]). Therefore the identification of human disease genes is in a way, equivalent to the identification of members of a pathway or process, which when mutated cause human disease. Genetic studies of heritable diseases using DNA linkage and association analyses yield genetic markers defining regions in the genome that may contain genes involved in the development of the diseases. PPI networks have been used in numerous studies to prioritize genes in these genetic loci, which often contain tens to hundreds of genes (some examples in [65-67]). A recent study showed that tissue relevant PPI networks built based on gene expression in the respective tissues performed better at prioritizing disease genes than the composite PPI network [68]. 'Guilt by association' methods have also been shown to help assign functions to proteins of unknown function that are connected in the PPI data to other proteins with known functions (for examples see [69, 70]). Network analyses of PDI have also been fruitful. PDI networks have been used to identify motifs such as feed-forward loops that are overrepresented and therefore biologically significant [54]. In PDI networks, network hierarchy has been recognized with TFs at different levels having distinct properties [53]. PDI networks have also been used to build predictive models for organism development [15]. In a *Drosophila* regulatory network study [51] that included TF-gene and RNA-gene interactions, it was shown that the

network was organized hierarchically with master regulators at the top of the hierarchy. It was shown that the TFs at the lower levels were frequently regulated by miRNAs. Overrepresented network motifs were also identified which showed that TFs combinatorailly regulated targets. This study also confirmed observations that TF-miRNA motifs may function to delay expression of targets, shedding light on mechanisms of how genes are regulated globally. Smaller sub-networks have been used in hypothesis-driven research to help identify the functional roles of genes based on the interaction neighborhood (for examples see [71-76]).

1.4 Global analysis of gene expression

Gene expression is the process of converting the information encoded in a gene or a transcriptional unit into a functional product such as a non-coding or regulatory RNA (ncRNA) or protein. The transcriptome is the complete set of transcripts in an organism, or in a particular biological sample. Global gene expression analysis is the simultaneous or parallel measurement of the transcriptome in many tissues or developmental stages, in contrast to measuring the expression of a single molecule or transcripts in a specific tissue or developmental time. Measurement of the transcriptome levels globally will provide a comprehensive picture of steady-state levels of transcripts. For protein coding genes the level of mRNA has been used as a proxy for the level of proteins. The quantities of a protein in a cell, however, can be regulated at other levels including during and after translation. A better measure of protein abundance may be the mRNA that is actually translated or ribosome-bound, while an even better measure would be the

quantification of the levels of proteins. However, currently the sensitivity and throughput of transcriptome analysis techniques far exceed those for proteomic analysis. In this section I am going to discuss how global gene expression at the level of transcribed RNA measured using microarrays and RNA-sequencing can be used to provide context to the interactome data and understand how the interactome is organized.

Different types of cells have varying complements of the proteome and ncRNA. Identification of global gene expression differences and comparative analyses between tissues or different treatment conditions or between healthy and disease conditions will provide insights into processes and pathways that are responsible for phenotypes or responses to treatments. Not only the identity but also the level of expression of genes is very important for tissue functions and phenotypes so it is important to do comparative quantitative analyses. Such global transcriptome analyses allow comparative analysis of gene expression in various contexts genome-wide. Recently, two global transcriptome studies were performed for all the tissues of *Drosophila* and also for different stages through the entire life cycle of *Drosophila*, providing insights into the biology of this model organism.

In the first study [7], Affymetrix single channel arrays with over 18,000 probes for close to 13,000 *Drosophila* genes was used to interrogate the transcriptome of all the adult tissues. This study showed that about half of the total transcriptome is expressed in each of the different tissues and that many genes are expressed predominantly in a single tissue. The authors also showed that the majority of transcripts that are expressed in the

embryo and important for development are also expressed in many adult tissues. It was further shown that several examples of genes with known functions have unexpected patterns of expression suggesting that they may have other novel functions. The study also identified novel transcriptional units for which probes were spotted on the array. The potential of *Drosophila* studies to elucidate the mechanisms of human diseases based on the fact that many human diseases gene orthologs were expressed in a tissue-specific manner was also addressed. The advantage of this transcriptome study is that the levels of expression of a gene in different tissues can be compared. An important part of the analyses included the demonstration that genes with restricted patterns of expression are better detected using tissue transcriptomics rather than whole organism transcriptomics. The drawback of the study is that one cannot compare the expression of two different genes as the data are semi-quantitative.

A second transcriptome study [4] identified new functional elements of the *Drosophila* genome using RNA sequencing (RNA-seq) to quantify transcript expression through the entire life cycle of the fly from early embryo to the adult spanning 30 developmental time points. They identified several thousand new functional elements in the genome including ncRNA, splice junctions, new genes and RNA editing events. The transcriptome was quantified using RNA-seq complemented by tiling microarrays. This project led to the identification of close to 2,000 newly transcribed regions of which a third could be new protein coding genes while the rest might be novel peptide or ncRNA transcripts. 37 new ncRNA species were identified along with 23 new primary miRNA transcripts. Contrary to previous estimates, over 40% of genes were found to be

alternatively spliced in different stages of development. Overall the project was able to detect and quantify the expression of 14,800 genes covering close to 90% of the annotated *Drosophila* genome. This study failed to identify about 1,500 known genes probably due to lack of tissue resolution as noted by the earlier microarray study [7]. This study showed that even for a well-studied model organism, many novel features can be discovered using RNA-seq. Since the same platform was used to measure transcript abundance, the expression levels of genes are comparable to each other. This study overcomes the disadvantage of microarrays in that the RNA-sequencing measurements are quantitative. The RNA-seq data from this study was integrated with TF binding data to infer TF regulation of gene expression [51].

As noted above, the available interactome data are composites of potential interactions in an organism as most interaction mapping studies do not take into account *in vivo* spatiotemporal context. The transcriptome studies discussed above have shown that a significant fraction of the transcriptome is expressed in a tissue or developmental stage specific manner. Genes have varying expression patterns; some genes are expressed ubiquitously and some have more restricted patterns of expression. Therefore, the transcriptome data may be used to identify networks of genes that are expressed in any given tissue or stage. There are challenges associated with using transcriptome data to filter interaction data. For example, it is difficult to decide based on global expression levels, the levels of expression that may be relevant in a particular tissue. I am going to discuss my studies on the correlations between the interactome and transcriptome correlation and between the levels of expression and mutant phenotypes in chapter 3.

1.5 Integration of heterogeneous data

Cellular pathways consist of networks of DNA, RNA, and proteins interacting with each other in many complex ways. For example, a biological process might include protein-DNA interactions, such as a TF binding to a gene (TF-gene) to regulate transcription, RNA-RNA interactions that regulate gene expression, and protein-protein interactions that form complexes to carry out a particular function. In order to get a system-wide understanding of a biological process, therefore, interaction networks must be built by collecting and integrating heterogeneous data such as protein-protein, protein-DNA and RNA-gene interactions. Networks generated by integrating such data can provide clues about the functions of individual genes and entire pathways [77-82]. Unfortunately, these different interaction data types are spread over many different databases making the processes of collection and integration non-trivial. Also, different genes are expressed in different cell types and at different time points and this in turn determines if an interaction can take place. Therefore interactome data must be integrated with gene expression data in order to build networks that are biologically relevant. In this section I am going to discuss the resources available to collect and integrate different types of data and their shortcomings, thereby showing the need for a one-stop resources for biologists.

1.5.1 Genomic databases

Genomic databases collect and curate data on the parts lists, namely genes. Some examples are FlyBase [83], WormBase [84] and NCBI [85] repositories among others [86]. These resources provide details of genes and their products including sequence,

regulatory elements, gene expression and other descriptive information. Some of these such as FlyBase [83] also provide controlled vocabularies or ontologies for function, phenotypes, anatomy and development. Genomic databases are therefore one of the resources a biologist has to consult to obtain gene-centered data.

1.5.2 Interaction databases

Interaction databases collect either PPI or PDI or other specific types of interactions. Some examples of large PPI databases include BioGRID [47], IntAct [48] and MINT [46]. The interactions in these databases include data from high throughput studies and manual curation of interactions from the literature. These databases have interfaces for visualization and for manual or programmatic access for data downloads. It is important to note that these databases are themselves not comprehensive and in order for a biologist to get a list of all known interactions, significant effort and time has to be spent collecting and integrating the interactions from all these databases [87].

PDI databases include REDFly [52] and data collected by the ENCODE [88] and modENCODE [89] projects. REDFly [52] has regulatory element data of high quality determined using DNase footprinting of TF binding sites. ENCODE [88] and modENCODE [89] provide interfaces to query the results of ongoing experiments for TF binding and other studies. However data showing that a TF binds to a gene under specific conditions may not necessarily mean that it regulates the gene.

Data available for RNA-gene interactions, specifically miRNA-gene interactions, for which data are available extensively in repositories such as TargetScan[90], MiRanda

[91], PicTar [92] and MinoTar [61]. These data also have to be downloaded and combined to get a comprehensive list of interactions as each resource has a different set of interactions.

1.5.3 Gene expression databases

Gene expression data are stored in many databases, for example GEO [93], FlyBase [83] and Flyatlas [7]. Proteomic data have been recently made available in a database [94]. Again, these databases have interfaces for querying, visualizing and download of the data. These data, however, have to be downloaded and processed before they can be integrated with other data such as interaction data, which is again time consuming and difficult.

1.5.4 Integration of heterogeneous data

As stated earlier, in order to get a more complete picture of a biological system, it is necessary to integrate all of the genome-wide data [95, 96]. In order to integrate the vast amounts of genome-wide data many computationally intensive steps have to be undertaken. Therefore, such integration is difficult for most biologists to perform, which usually leads to insufficient use of the massive amounts of useful data that are available. There is a need for databases to take up this task of collecting all the interaction data into one resource and providing tools for its analysis [97]. One such database resource, and the only one of its kind, is DroID (http://droidb.org/), the *Drosophila* Interactions Database, which seamlessly integrates various types of interaction data into one resource (Figure 1-1) and provides intuitive genomic and expression filters to obtain subnetworks of interest [98]. DroID is a comprehensive public resource for *Drosophila* gene and

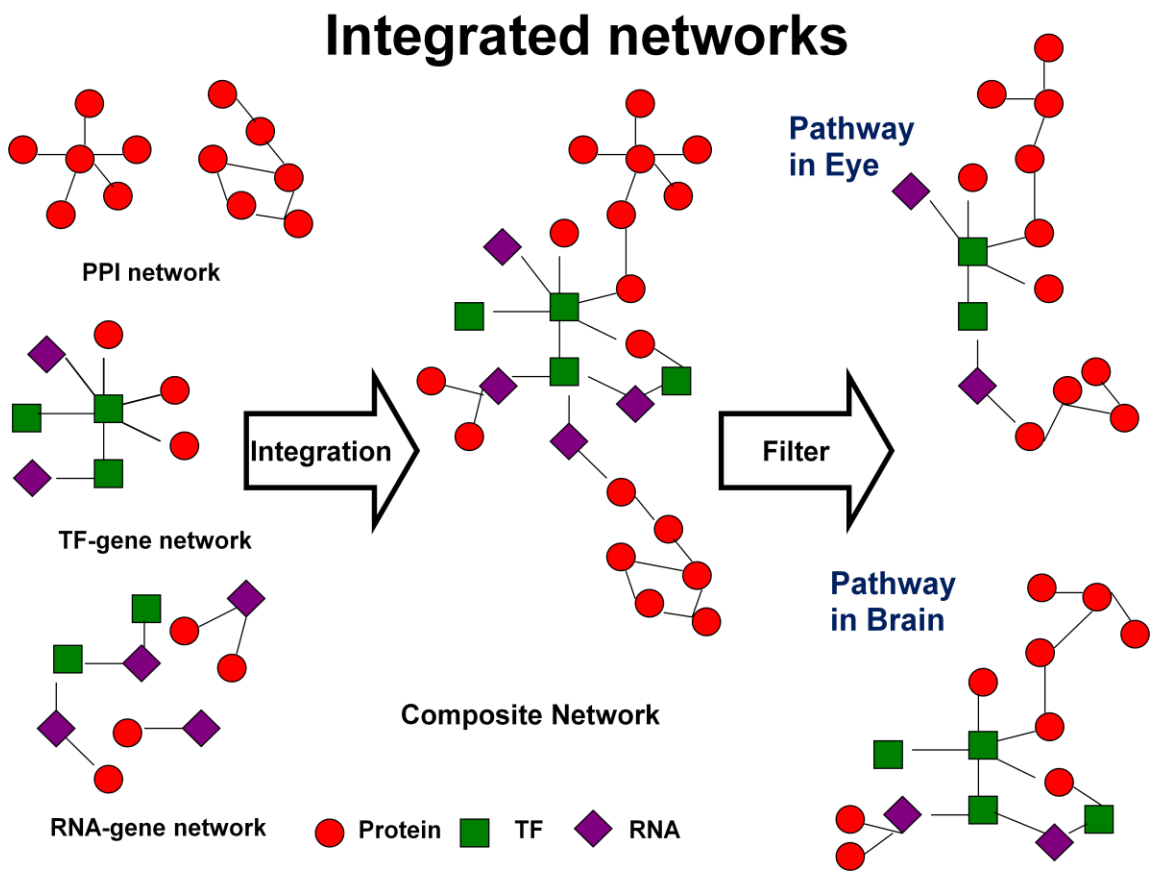


Figure 1-1. Integrated networks. DroID includes PPI, TF-gene and RNA-gene interactions that can be easily integrated to make more complete biological networks. The networks can be filtered using gene expression filters available in DroID to obtain networks that are relevant in different contexts.

protein interactions. DroID contains experimentally detected PPI curated from the literature and from external databases, and predicted protein interactions based on experiments in other species. I set out to expand DroID to include heterogeneous interaction types that would enable more complete analyses of the genetic networks that underlie biological processes. As described in chapter 2, in addition to PPI and genetic interactions, the database now includes TF-gene and regulatory RNA-gene interactions. In addition, DroID has gene expression data that can be used to search and filter interaction networks.

1.6 Project Outline

The two broad aims of my project include:

1. To integrate data on various cellular networks and provide ways to obtain tissue or stage relevant networks.

2. To discover organizing principles of biological systems and use the principles to provide ways to use the vast amounts of genome-wide data in an unbiased and systematic way to generate useful information.

Accordingly, in Chapter 2 I discuss DroID, which is a database with tools to integrate and visualize as well as download the integrated interaction data. I first introduce DroID and then discuss the enhancements I have made to make it a more useful analysis tool for biologists.

In Chapter 3, I integrate interaction and expression data to find out how the interactions between genes that are widely or specifically expressed differ. In light of the results from this analysis, I developed a method to filter interaction networks based on normalized gene expression and I validate the method by showing that the filtered networks are enriched for biologically relevant information.

In Chapter 4, I discuss the future of DroID and also the future for research on biological systems in light of the fact that data generation and technology development are both in exponential phase while methods for making sense of the data and gaining biological insights are lagging far behind.

CHAPTER 2 THE *DROSOPHILA* INTERACTIONS DATABASE

Most of the work described in this chapter has been published in **Nucleic Acids Research. 2011;39(Database issue):D736–43. PMID: 21036869**

2.1 Introduction

Networks of interacting genes and gene products perform various cellular functions. In order to get a system-wide understanding of biological processes, therefore, interaction networks must be comprehensive, including all the available interaction data. Published literature curated protein-protein interaction (PPI) data have been available for different organisms from several large centralized repositories such as MINT [99], BioGRID [100] and IntAct [101]. The scope and content of these repositories, however, have been highly variable and each database has some unique interactions not found in the others. To get a complete set of interactions, therefore, data from all these sources have to be integrated, which is a non-trivial task [102, 103]. Another shortcoming of these databases is that they do not have organism-specific data such as phenotypes, expression data, organism-specific vocabularies, or alternative identifiers. Organism-specific data are necessary to provide context to interaction data. It is also difficult to identify conserved interactions or interologs in these databases. In order to address these problems the *Drosophila* Interactions Database (DroID; www.droidb.org) [87, 104] was built to be a comprehensive protein interaction database that also stores genetic interactions and interolog data specifically for the model organism *Drosophila*. DroID addresses some of the shortcomings of other PPI databases by combining PPIs from all

the available sources and interolog interaction data. The data can be browsed and retrieved via easy-to-use interfaces [104], including a web interface to search interactions, a Java applet called Interaction Browser (IM) to visualize interactions, and a DroID Cytoscape plug-in for the network visualization and analysis program, Cytoscape [105]. DroID also provides organism-specific information along with interaction data in a one-stop resource. The interaction data can also be filtered using confidence scores for PPI [106], providing measures to estimate the biological relevance of specific networks.

Although DroID as originally described [87, 104] had many advantages over other PPI databases, it also shared some of their shortcomings. It only included PPI and genetic interactions, while a biological pathway would most likely include both TF-gene and RNA-gene interactions along with PPI. In order to get a complete picture of a biological pathway, other types of interaction data needed to be integrated into DroID. For example, a biological pathway may include protein-protein interactions at the membrane that initiate a cascade of signals transduced by other PPI that in turn activate transcription factors. Transcription factors may then bind to and activate or repress the transcription of specific genes. Gene expression can also be controlled by regulatory RNAs or non-coding RNAs (ncRNAs), such as microRNAs (miRNAs or miRs). microRNAs post-transcriptionally regulate the expression of specific genes by base pairing with target mRNAs [58]. Several studies have shown the value of integrating different types of interaction data (for recent examples see [77-82]). Just like the PPI data, data for other interaction types are spread over many different databases making the processes of collection and integration difficult. In this chapter and in [98], I describe an

overhaul to DroID that addresses these problems by collecting and combining TF-gene and RNA-gene interactions along with the PPI into a single resource where they can be analyzed together.

Another major shortcoming of PPI databases including the original DroID was the lack of extensive gene expression data to provide biological context. An important feature of the interaction network generated from interaction data is the lack of spatiotemporal context. For example, PPIs detected by yeast two hybrid or by expression of affinity tagged bait proteins do not capture the spatiotemporal variation in expression of genes in a living organism. Instead, the data are a composite of potential interactions in an organism. *In vivo*, a gene may have one set of interaction partners in one cell type and a different set in another cell type or at a different time (example in [107]). In one view, the interaction network defines the cell type and its developmental stage. In order to obtain a meaningful picture of a biological process then, it is necessary to integrate spatiotemporal gene expression patterns with the interaction data.

In order to address these issues DroID was upgraded in 2010. As part of the upgrade, I added new heterogeneous interaction types including TF-gene and RNA-gene interaction data. I also added more gene attribute data including phenotype and expression terms. I also added new gene expression data sets to enable users to generate tissue or stage relevant subnetworks. In the following sections I detail the improvements to DroID including enhancements to gene information, improved PPI data, addition of new types of interaction data and addition of the recent genome-wide tissue and stage

wide gene expression data, moving DroID from a database for PPI to a more general tool for analysis of gene regulatory networks.

2.2 Gene information

DroID currently has interaction data for over 15,000 genes, including more than 13,785 protein-coding genes (Table 2-1). Most of the gene information comes from Flybase [108], the authoritative resource for *Drosophila* genome information. Each gene page of Flybase.org has a link to DroID, which takes users to the interaction summary page for the gene. DroID interfaces in turn allow searches of all Flybase gene attributes located in routinely updated DroID tables or via links to Flybase [109]. DroID uses Flybase gene identifier numbers (FBgn) as the primary identifiers for genes. A typical search starts with finding genes of interest in DroID using gene names or identifiers, or gene attributes such as Gene Ontology terms [110, 111], gene class, and newly added phenotype and expression terms (Figure 2-1). Users can then view the interactions for the genes in the form of a list or build an interaction map. The gene information and other attributes can also be used to filter interaction data or to manipulate the data in an interaction map.

Another new feature of DroID is gene ortholog information, extending the usefulness of the database beyond the community of *Drosophila* biologists. A large number of mapped gene and protein interactions are available for *Drosophila*. This fact combined with the conservation of genes and pathways between *Drosophila* and other multi-cellular organisms (including humans), suggests that scientists studying genes and

pathways in other organisms could gain insights from the data contained within DroID that may not be available for the organism they are studying. To facilitate the use of DroID by scientists studying other organisms, the upgraded DroID contains gene names and identifiers for the *Drosophila* orthologs of human genes and genes from other model organisms including yeast, worm, zebra fish, frog, and mouse. These mappings, which are based on the InParanoid [112] orthology mapping algorithm, thereby expand the number of different gene name synonyms or identifiers that can be used to search for *Drosophila* genes. For example, users only familiar with the human or worm name of a gene could readily search DroID for interactions involving the *Drosophila* orthologs (Figure 2-1).

A more recent addition to the gene attributes included the addition of RNAi data for more than 2,600 genes based on a study from the Perrimon laboratory [38] (details in section 2.3). Along with this, the gene orthology information was expanded to include human disease gene information from the OMIM database [113].

2.3 Protein-protein interactions

2.3.1 Improvements to existing data sets

A large amount of PPI data for *Drosophila* comes from large-scale yeast two hybrid studies, which together have combined to identify >24,000 interactions [34-36]; these are stored as three different data sets in DroID as each of them include study-specific details. I have also included >5,000 unpublished *Drosophila* PPIs as they are being generated in an ongoing high throughput yeast two hybrid screen

Table 2-1: Summary of DroID v2013_02 Data (updated 1 Feb 2013)

Data set	# interactions	# genes
PPI: Curagen yeast two hybrid	1,9559	6,692
PPI: Finley Lab yeast two hybrid	9,019	3,618
PPI: Hybrigenics yeast two hybrid	1,843	1,270
PPI: Perrimon Lab co-AP/MS	3,84	252
PPI: DPIM co-AP/MS	61,312	4,984
PPI: From other major databases	3,720	2,074
PPI: Human interologs	68,137	4,965
PPI: Yeast interologs	80,127	2,782
PPI: Worm interologs	2,989	1,690
PDI: Transcription Factor-gene	157,769	12,353
RRI: miR-Gene	113,549	12,386
GI: Genetic interactions	7,166	2,895
Total number of unique interactions	513,357	15,194

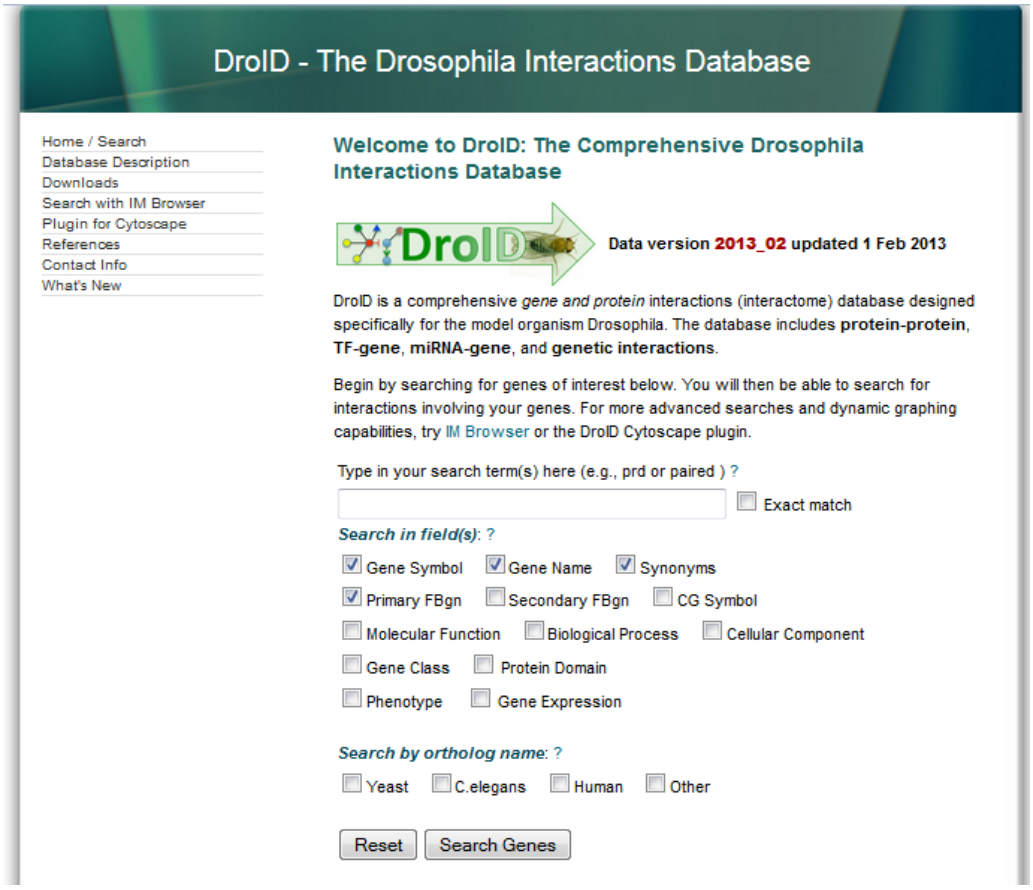


Figure 2-1. The DroID gene search page. Genes can be searched using data from the gene attributes table such as Gene Ontology terms [110, 111], gene class, phenotype and expression terms and orthologs.

(www.proteome.wayne.edu). I have also included all of the *Drosophila* PPI data from the literature-curated multi-organism databases, BioGRID, IntAct, MINT and BIND [46-48, 114]. Each of these databases has a unique set of interactions not found in the others (Figure 2-2) making DroID the most comprehensive single resource to analyze all the available PPI. The PPI data includes about 900 new interactions from the BIND database [114] added to DroID 2012_04 version; ~550 of these were unique to this resource. About two thirds of the ~900 interactions were mappable programmatically to FlyBase symbols and I curated the rest manually to ensure minimal loss of data. BIND is no longer updated, so this was a one-time addition to DroID (Figure 2-2). In contrast, BioGRID, IntAct and MINT are frequently updated and new data from these databases get incorporated into DroID with each DroID update.

2.3.2 New coAP complex data sets for *Drosophila*

A large-scale co-affinity purification (co-AP)/Mass Spectrometry (MS) screen was performed in the Perrimon laboratory [38] using 15 tagged bait proteins and interacting proteins in the complexes were identified. C-terminally TAP-tagged proteins were expressed in stably transfected S2R+ cells, complexes were affinity purified, and the associated proteins were determined by liquid chromatography followed by tandem MS (LC/MS/MS). The screens were done under different conditions (i.e., presence or absence of EGF or insulin signaling), giving the data a dynamic component. Parallel RNAi screens were also performed to identify pathway components in different conditions. Working with the Perrimon laboratory, I have added this PPI data set of 384

interactions among 252 proteins to DroID, so that it can be analyzed separately or in combination with other data sets. I also added the RNAi screen data for about 2,600 genes under different assay conditions to the gene attribute table. We described this update to DroID in [38].

In another recent study [37], the *Drosophila* Protein interaction Map (DPiM) project has generated a large amount protein complex data for *Drosophila*. In this study, close to 3,500 C-terminally FLAG-HA-tagged bait proteins were expressed in transiently transfected S2R+ cells, the tagged bait proteins were affinity purified, and the interacting proteins were identified by LC-MS/MS. The interactions in this data set also include study-specific confidence scores. This data set includes more than 60,000 interactions for close to 5,000 proteins. Other study-specific details such as isoform information and the number of baits a protein came down with have also been included in DroID.

The protein complex data from both coAP studies have been converted to binary interactions using the hub-spoke model where each bait is predicted to interact with all of the co-purified proteins. This data along with the yeast two hybrid data should enable identification of more complete pathways as the data sets are complementary to each other.

2.3.3 Interolog Interactions

About 60% of PPI in DroID are interologs predicted from yeast, worm and human. Interologs have been predicted based on the premise that if proteins are conserved across species then they could have potentially conserved functions and there-

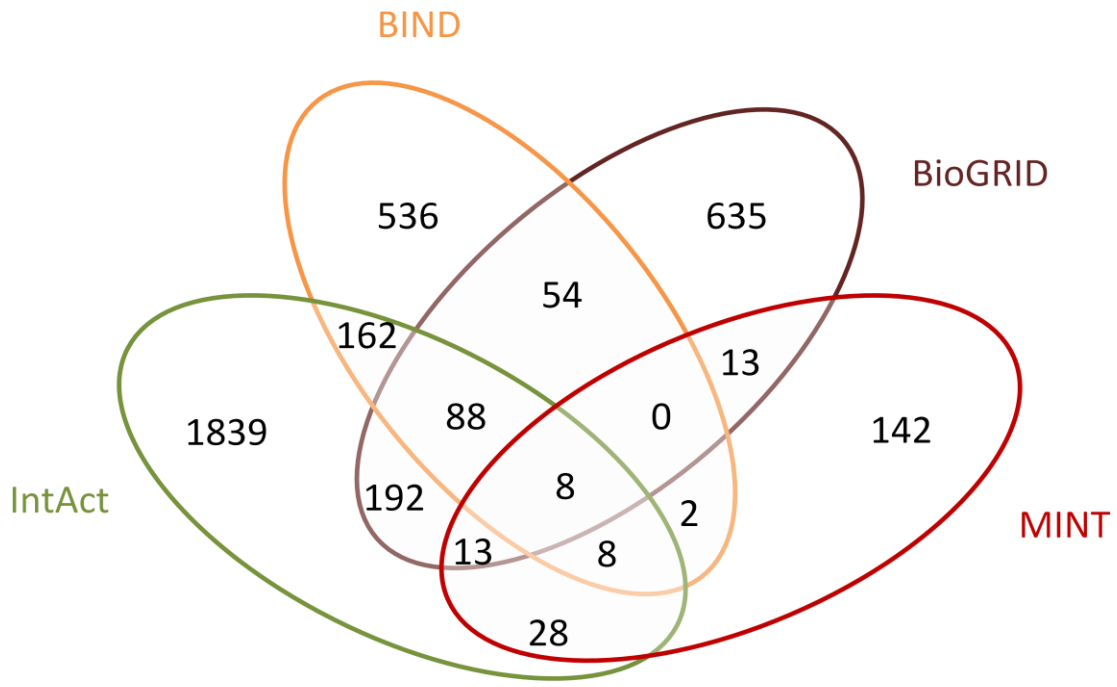


Figure 2-2. *Drosophila* protein-protein interactions in four major online databases and stored in the table 'PPI from other major databases'. Numbers reflect the data publically available as of February 2013. DroID collects literature curated PPI from all four databases that are not part of the large scale yeast two hybrid or coAP complex studies that are separately included in DroID.

fore also have the same interaction partners across species [115-117]. Interolog data in DroID are generated by combining PPI from BioGRID, IntAct, and MINT for *S. cerevisiae, C. elegans* and human [46-48]. For human, data from Reactome [118] and HPRD [119] are also included. Potential *Drosophila* orthologs of yeast, worm and human proteins are identified using data downloads from the InParanoid database [112]. The interacting proteins are then mapped to *Drosophila* orthologs to generate predicted conserved interactions.

Recent studies have explored the extent of conservation of predicted interactions. In one study it was suggested that only ~ 11% of interolog interactions might be conserved between yeast and human based on homology and the rate of loss of interactions as species evolve [120]. Another conclusion from that study was that transfer of interactions to evolutionarily distant species might be more reliable than to species that are closer evolutionarily such as between two yeast species. In other studies the rate of conservation of PPI in complexes versus conservation of other interactions was considered. It was suggested in these studies that interactions among members of the same complex may be more conserved than the interactions between complexes [121, 122]. From these studies one may conclude that certain regions of the interaction network are more conserved across species than others. Therefore it is important to keep in mind the source organism data for interolog predictions as well as the type of interactions (within complex versus between complex) that are predicted when using the interolog data for studies in *Drosophila*.

Interolog and other data can also be used to assign confidence scores, probabilistic scores that show how likely it is that an interaction is a true positive. One confidence scoring method [123] utilizes several different training data sets, one of which is a set of interactions conserved in more than one organism. These scores are available in DroID. Also, in a recent large scale protein mapping study [37], a high confidence interaction for *Drosophila* was deemed as one that was found in more than one species. An earlier study [124] was able to validate about 50% of their interolog predictions, which were based on three different model organism networks, highlighting the value of interologs. It was also shown by our group that conserved PPI had significantly higher gene expression correlations than random pairs of genes [87].

2.4 New types of interaction data

2.4.1 Protein DNA interactions

The upgraded DroID now includes a new table for TF-gene interactions that can be searched separately or combined with the PPI and genetic interaction data. The TF-gene interaction table includes data from REDfly [125], a continuously updated database of interactions curated from published literature. DroID also now includes TF-gene data from the modENCODE project [51]. The TF-gene table now holds ~150,000 interactions between 149 TFs and ~12,000 targets. The high-quality data from REDfly includes interactions demonstrated experimentally by DNaseI footprinting assays. The modENCODE project used a machine learning algorithm to infer interactions between a TF and its targets. The algorithm took as inputs TF-binding data from ChIP-chip and

ChIP-seq studies (both from the modENCODE project and from other published work) and expression data [4]. The modENCODE project used REDfly data as training data for their machine learning algorithm. I have merged these two resources to make a list of unique gene-centered interactions. Consistent with the philosophy of DroID, I have included all information that is available for these interactions including links to the original source publications and to the source pages. Although the finding that a TF binds and regulates a gene under specific conditions does not necessarily mean that the TF regulates the gene under all conditions, these data do provide evidence that a TF may regulate the gene.

2.4.2 RNA gene interactions

In contrast to the vast amounts of TF-gene interaction data, presently there are very few experimentally verified RNA-gene interactions for *Drosophila*. Most sources of RNA-gene interactions derive from predictions that use base-pair complementarity between miRNAs (miR) and their putative targets [60]. In *Drosophila*, over 140 miRs have been identified thus far and more are likely to follow [126]. The upgraded DroID contains predicted miR-target gene data for 146 miRs belonging to 122 miR families from the TargetScanFly [90] database. I have mapped the miR identifiers to Flybase gene identifiers using information available at Flybase and MirBase [127]. I have also added RNA-gene interactions predicted from MinoTar [61] and interactions from the modENCODE project [51]. I have included all information available for these interactions including information on whether the prediction is for conserved miR

families or non-conserved miR families, the target site conservation, and the original prediction database. These data can be easily combined with the other types of interaction data for further analysis and visualization in interaction networks.

2.5 Filters for interaction data based on gene expression

I have worked to include four different types of gene expression data to enable users to filter interactions based on gene expression patterns, which are described in the sections below.

2.5.1 Genome wide gene expression data

I have included raw gene expression data in DroID. One data set was derived from a microarray study on embryogenesis that examined RNA expression at different developmental stages [128]. I recently added stage-wide temporal gene expression data from Graveley *et al*. [4]. This data set includes the mRNA abundance determined by RNA-Seq for about 14,000 genes spanning 30 developmental time points from embryo to adult. These two raw data sets can be used to filter interactions based on any user-defined cutoff.

2.5.2 Normalized gene expression values

I created normalized gene expression data tables based on gene expression data for tissues and stages. The tissue-wide gene expression data set is from flyatlas.org [7]. This data set includes the mRNA signal that correlates with mRNA abundance for about 13,000 genes in 15 adult and 8 larval tissues measured using Affymetrix *Drosophila* expression arrays. The stage-wide temporal gene expression data set was obtained from

Graveley *et al*. [4]. The normalized expression of a gene is calculated as the percent of its maximum expression level across all tissues or stages. For example, if a gene is maximally expressed in the ovary compared with all other tissues, then its normalized value in the brain is its expression in the brain as a percentage of its expression in the ovary. I have shown that applying this filter to interaction networks enriches for context relevant interactions. Further details and validation of this filter are discussed in Chapter 3.

2.5.3 Gene expression correlations

Gene expression correlation values indicate the level of similarity in the expression patterns of a pair of genes. Gene pairs that are upregulated or downregulated similarly across many tissues or developmental stages have higher expression correlations than genes that are not frequently expressed together. Previously, our group showed that genes encoding interacting proteins are more likely to have higher expression correlation values [87] and that expression correlation can be used to assign confidence to PPI [106]. Originally in DroID the correlation values were computed based on the extensive genome-wide expression data available in the NCBI Gene Expression Omnibus (GEO) [129]. This resource has a wide variety of data sets measuring expression in different conditions, including many that may not be physiologically relevant. After the tissue-wide [7] and stage-wide [4] expression data for *Drosophila* under normal physiological conditions were made available, I used these data for computing gene expression correlations. I have computed ~0.2 billion correlations for all gene pairs in each of the tissue-wide and stage-wide gene expression data sets. Since

these correlations are computed from the physiologically relevant data sets they are potentially more useful. As DroID collects more interaction data, the precomputed correlations can be used immediately during every database update. The correlation values can be used to filter interaction data focused on networks of genes that are expressed together and therefore more likely to function together. The correlation values for interacting pairs are also useful for a variety of other analyses and are available for download at the DroID website.

2.5.4 Qualitative expression data

DroID now has a qualitative expression filter based on spatiotemporal RNA and polypeptide expression and localization data curated by FlyBase [83]. The data are from published literature and are annotated using controlled vocabularies. During every DroID update these data are downloaded, processed and stored in the gene attributes table. Along with the already existing expression filters in DroID, the localization information will help biologists to better explore the dynamic aspects of interaction networks.

2.6 Implementation

In order to keep pace with the vast amounts of newly generated PPI and other interaction data, DroID is regularly updated. I have implemented a streamlined protocol for updating the gene and interaction data. The update protocol combines automatic collection of data from external databases with intervening steps involving limited manual curation of the collected data sets and high throughput data. A complete update of the database can now be completed in less than one day, which allows for an improved

update schedule. The three major steps in order of execution are collecting gene attribute data, collecting interactions from other repositories and updating the oracle database followed by a final manual verification of the data at the user interface.

FlyBase [83] provides public read-only access to their Chado [109] database. This is used to first programmatically download gene attributes, expression, and genetic interaction data and then combine and process the data into a format that is easily uploaded into the DroID Oracle database in a single step (scripts in Appendix A). Gene mappings are also downloaded and processed from FlyBase as a) gene splits and mergers are a common annotation correction and b) each of the interaction data sources mentioned below uses different gene identifiers. Gene orthologs are generated using downloads from Inparanoid [112] database (scripts in Appendix A).

Apart from the PPI from large-scale yeast two hybrid and coAP complex studies resident in the database, DroID also obtains and includes all of the *Drosophila* PPI data and interolog interaction data (yeast, worm and human) from the literature-curated multi-organism databases, BioGRID[47], IntAct [48] and MINT [46] along with Reactome [118] and HPRD [119] for human. BioGRID[47], IntAct [48] and MINT [46] provide a standard interface (PSICQUIC - http://code.google.com/p/psicquic/) to access their interaction data programmatically. Using the RESTful (Representational state transfer) method and the common query language (MIQL- The Molecular Interactions Query Language (MIQL) provided by the PSIQUIC service, the interaction data are first obtained and the gene mappings from FlyBase and the orthologs' identifiers are used to generate uniform gene identifiers (FBgn#s) for the interaction data; again the data are

combined and processed such that they can be uploaded into DroID in a single step (scripts in Appendix B).

After the tables have been updated with the new data, all the gene identifiers in DroID are updated to the latest annotation of the genome available in FlyBase by scripts run directly on the newly combined Oracle database, again using the latest gene mappings downloaded and processed from FlyBase. This process may result in a reduction of interactions as an old identifier split into two new primary identifiers would be removed (scripts in Appendix C).

Once all the data have been uploaded and the gene mappings have been updated, a final step is spot checking the data at the user interface to make sure all the fields are populated. Also any records rejected by the Oracle database engine have to be manually checked and processed before upload into DroID. Finally, the number of interactions in the previous version and the current version are compared to make sure the data collection and upload was successful. This step also includes making the downloadable files.

2.7 Summary

DroID was originally created to serve as a repository for all *Drosophila* PPIs. Our understanding of the importance of integrating additional interaction types along with PPI led to the evolution of the database. DroID now serves not only as a comprehensive resource for *Drosophila* PPI data but also as a tool for combining PPI data with TF-gene, RNA-gene, and gene-gene data to construct more complete biological networks. DroID is the only database that allows the integration of all these types of heterogeneous

interaction data. The combination of the multiple interaction types, the spatiotemporal gene expression data, and intuitive filters based on quantitative genome-wide gene expression allows users to explore interaction networks that are relevant to specific developmental times and tissues. As the amount and diversity of interaction data continues to increase, databases such as DroID that allow users to access and interpret comprehensive gene and protein interaction data will become essential tools for the study of biological pathways and networks. That DroID has been an important community resource is evidenced by the fact that it has been used extensively to validate a scoring scheme for large scale protein complex interaction mapping study including ~5,000 bait proteins [37]. This study used DroID data to also assess the coverage of their study by checking what percentage of high confidence interactions in DroID were recovered in their network mapping study. Apart from this, in the two years since the updated DroID has been available numerous small scale studies and large scale systems biology studies have been performed based on the data sets available in DroID. For example, the heterogeneous data types in DroID were used in several hypothesis-driven studies [71-76], TF-gene data in DroID were used to evaluate predictive models in systems biology research [130-133], all of the heterogeneous interaction types were analyzed together [134, 135], used in studies on evolution [136-138], and in a genome-wide association study [139] in *Drosophila*. DroID data are also used by other bioinformatics tools [140] and cited as an important resource for interactions [47, 141, 142].

CHAPTER 3 INTEGRATING THE INTERACTOME AND TRANSCRIPTOME OF *DROSOPHILA*

3.1 Introduction

Functional genomics studies have shown that different cell types or tissues express different sets of genes [4, 7, 143-147]. It follows from these studies that the phenotypic identity of a cell or a tissue is governed in part by the particular collection of genes that are expressed in it. The different cell types or tissues, therefore, are likely to contain different regulatory networks of interacting genes and proteins. This is a primary motivation for integrating expression data with interactome data. Most of the available interaction data, however, has been derived from methods that do not identify the *in vivo* spatial or temporal context of the interactions [24, 148].

PPI networks have been built using interactions detected by high throughput methods for yeast [28-32], worm [33], fly [34-38], and human [39-41] and manually curated low throughput interactions [46-48]. The majority of available PPI have come from two methods, yeast two hybrid [42] and protein complex determination [43]. In the yeast two hybrid system, interactions between pairs of proteins expressed in a yeast nucleus are detected using reporter gene expression assays. In complex pull downs, a tagged protein expressed in cultured cells and all the proteins that interact with it are pulled down (purified) together and identified by mass spectrometry. The sequence to which a TF binds along with expression of the target in artificial conditions is used to infer a regulatory interaction between a TF and its target gene and build PDI networks

[51-54]. In ChIP assays, a tagged TF is expressed in cultured cells and the TF-bound DNA is isolated and the sequence is identified and mapped to the genome. Protein-DNA interactions (PDI), and in particular transcription factor (TF) binding sites, are also being mapped by methods such as chromatin immuno-precipitation (ChIP) or DNA protection assays [3, 51, 52]. The resulting interactome networks are thereby composites of hundreds of thousands of potential interactions that can take place in an organism. Many genes are expressed differentially in different developmental stages (temporally) and tissues (spatially) of an organism and this in turn determines if a potential interaction can occur *in vivo* [4, 7, 149].

In order to understand the organization of the PPI interactome networks based on gene expression patterns, several studies have focused on the tissue-specificity of gene expression. For example, it was shown in human studies [150-152] that tissue-specific proteins had very few interactions, most of which were with the house-keeping or ubiquitous proteins. In one study [150], a network of experimentally derived human PPI from public repositories was assembled for about half the of the human proteome and correlated with qualitative microarray gene expression data for many human tissues and cell lines. Subsequently, using the same human PPI network, another study was performed [151] correlating the more recent RNA-seq expression data for a few tissues and a few cell lines. The novelty of the second study was that gene expression data was used quantitatively, as the expression across different tissues was comparable. The study confirmed that tissue-specific proteins had few interactions in the PPI network and additionally found that the majority of the interactions in the network occurred

universally [151]. A third study used a network of curated human PPI and derived tissue networks based on gene expression. That study found that house-keeping genes had more interactions among themselves while tissue-specific proteins predominantly interacted with the house-keeping proteins rather than among themselves [152]. If this were a general principle, filtering interaction data based on tissue-specific expression patterns would not be effective at identifying tissue-relevant subnetworks. These studies suggested that tissue-specific proteins primarily interact with the more conserved ubiquitous proteins to modulate tissue-specific functions, but this has been shown thus far only with human proteins and with limited quantitative tissue expression data.

A series of other studies have focused on integrating different types of expression data with PPI interactome networks with the aim of increasing the accuracy of predictions of genes responsible for specific phenotypes or diseases. In these studies, tissue relevant networks were built by integrating interactome data with expression data to enable better prioritization of genes relevant to the human tissue function and disease [68, 153], for predicting tissue relevant phenotypes in mouse [154] and in studies of human tissue specific pathways [155, 156]. In all these studies however, expression data were used qualitatively (on-off calling or thresholding) to obtain tissue relevant networks.

In this study I first set out to determine how interactome data correlates with gene expression data in *Drosophila*. I took advantage of the extensive PPI and PDI interactome data available for *Drosophila* [98], and recent genome-wide, organism-wide tissue [7] and stage [4] transcriptome data. As suggested by the studies with human PPI data, I found that for *Drosophila,* tissue-specific proteins frequently interacted with ubiquitous

proteins and also that tissue-specific proteins had few interactions among themselves. In addition, I show that stage-specific proteins had few interactions among themselves but frequently interacted with the ubiquitous proteins.

Geneticists have long known that the level of expression of genes is an important factor in phenotypic variation. The problem of filtering a network based on quantitative gene expression, however, is particularly challenging when the range of gene expression values are considered. For example, the level of mRNA abundance in one study [4] had a range of several orders of magnitude between genes and a similar range of mRNA signal was observed in another study [7]. This problem, along with the interactome-transcriptome correlation I found, led us next to devise a normalization procedure for gene expression data that would take into account the level of expression. The normalized expression value for a gene in a tissue or developmental stage is the percentage of its maximum expression level. In order to identify networks that operate in different contexts, the composite interactome networks were filtered using this intuitive and quantitative gene expression filter. I show systematically that these filtered subnetworks are enriched for phenotypes that are relevant in different stages and tissues.

3.2 Materials and methods

3.2.1 Interaction and expression data

Protein-protein interaction (PPI) data were downloaded from DroID [87, 98] version 2011_05, and include 93,544 experimentally detected PPI from yeast two hybrid data from three large-scale studies [34-36] and an ongoing project [98], literature curated PPI from other major databases [46-48], and recently added PPI data from two large co-

AP complex studies for *Drosophila* [37, 38]. The PPI data also include 144,171 *Drosophila* interologs predicted from experimental data in yeast, worm and human [87, 98]. In total I analyzed 235,950 unique PPI involving proteins from 10,823 genes. The *Drosophila* PDI data include 158,508 unique regulatory PDI for 149 transcription factors (TF) and 12,441 of their target genes that were inferred using TF binding and correlated expression of targets [51, 52]. The tissue-wide gene expression data were downloaded from Flyatlas.org [7]. These data include the mRNA signal that correlates with mRNA abundance for about 13,000 genes in 15 adult and 8 larval tissues measured using Affymetrix *Drosophila* expression arrays. The stage-wide temporal gene expression data were obtained from Graveley *et al*. [4]. This data set includes the mRNA abundance determined by RNA-Seq for more than 14,000 genes spanning 30 developmental time points from embryo to adult.

3.2.2 Gene expression specificity scale

Genes were classified according to the specificity of their expression levels across all tissues or developmental stages. The expression specificity (Es_i) of a gene in tissue or stage i is simply a fraction of the total abundance across all tissues or stages and was calculated as follows:

$$Es_i = \frac{abundance_i}{\sum_{i=1}^{n} abundance_i}$$

where $abundance_i$ is the raw expression value of a gene in tissue or stage i. This results in Es_i values between 0 and 1 for each gene in each tissue or stage. The sum of all expression specificity values for each gene across all tissues or stages equals 1.

I placed genes into three nonoverlapping bins based on their specificity of expression across all 15 adult tissues: Genes with Es_i values of >=0.8 in any of the adult tissues were labeled as tissue-specific (2,838 genes); genes that were not tissue-specific and that had non-zero expression values across all adult tissues were labeled as ubiquitous (3,960 genes); the remaining genes were labeled as tissue non-specific-non-ubiquitous (5,830 genes). For the analyses in Figures 3-5A and 3-5B, I similarly classified genes using only the 8 larval tissues. In a separate classification, I placed genes into three non-overlapping bins based on their specificity of expression across 30 developmental stages: Genes with developmental stage Es_i values of >=0.19 were labeled as stage-specific (3,566 genes); genes that were not stage-specific and that had developmental stage Es_i values >0.005 across all time points were labeled as ubiquitous (4,972 genes); the remaining genes were labeled as stage non-specific-non-ubiquitous (6,064 genes). The stage-specific bin included genes that showed a transient and sharp change in abundance that spanned a maximum of four consecutive developmental time points in a majority of the cases. The overlap among genes classified by the two specificity scales (tissue specificity and stage specificity) is shown in Table 3-1.

3.2.3 Percent of maximum expression level scale

I created a normalized gene expression scale to indicate for each gene the extent of its expression in a particular tissue or stage relative to the maximum level of expression it displays in any tissue or stage, respectively. I refer to this normalized expression scale as the percent of maximum or pmax. To normalize gene expression based on this scale, the expression of a gene is calculated as the percent of its maximum

Table 3-1 Overlap of different groups of genes binned based on expression. Overlap of genes between each of the stage expression based bins and tissue expression based bins. Numbers in parentheses are total number of genes in each bin.

	Adult tissue-specific (2,838)	**Adult tissue-NSNU (5,830)**	**Adult tissue-ubiquitous (3,960)**
Stage-specific (3,566)	1,363	1,546	41
Stage-NSNU (6,064)	1,353	2,419	536
Stage-ubiquitous (4,972)	49	1,290	3,226

expression level across all tissues or stages. For example, if a gene is maximally expressed in the ovary compared with all other tissues, then its pmax value in the mid-gut is its expression level in the midgut as a percentage of its expression in the ovary. A gene is expressed in tissue or stage *i* at a percent ($pmax_i$) of its value in the tissue or stage where it is maximally expressed. $pmax_i$ is calculated by:

$$pmax_{\mathrm{i}} = \frac{abundance_i}{abundance_{max}} X\ 100$$

where $abundance_i$ is the raw expression value in tissue or stage *i*, and $abundance_{max}$ is the raw expression value in the tissue or stage where it is maximally expressed. Two overlapping groups of genes were analyzed: those with $pmax_i$ >50% and those with $pmax_i$ >75%. Filtered gene lists and networks were built from these two filters for each tissue or stage.

3.2.4 Orthology mapping

Potential *Drosophila* orthologs of yeast, worm and human proteins were identified using data downloaded from InParanoid version 7.0 database [112]. InParanoid performs pairwise comparisons of proteomes and constructs orthology groups. An orthology group has at least one protein from each species (seed orthologs) that are more similar to each other than to any other sequence in the other proteome. The orthology group may have additional sequences that are closer to the seed orthologs than to any sequences in the other proteome. I used PERL to merge the InParanoid groups keeping a *Drosophila* gene as a unique reference to each orthology group.

3.2.5 Network analyses

To compare the numbers of interactions among different groups of genes I calculated the fold difference in interactions over random expectation. First I counted the number of interactions among genes in the test set. Then I picked the same number of random genes from the PPI network minus the test set and counted interactions among them. The fold difference is the number of interactions among the test set divided by the number of interactions among the random set. I repeated this 5,000 times for each test set. To calculate a p-value for each test case, I performed 100,000 Monte Carlo simulations by picking gene sets of the same size randomly and counting the number of interactions between the genes in the random sets. I computed the number of times the interactions in the random sets were as low or lower, or as high or higher, depending on the case. This number was used to calculate the binomial confidence interval using binom.confint in R to calculate the p-value at a confidence of 99.99% (CI = .9999) [157, 158]. I used the upper confidence interval as the p-value.

3.2.6 Enrichment analyses

Gene Ontology enrichment analysis was performed using DAVID 6.7 [159]. I used DroPhEA [160] and BiNGO [161] to perform phenotype enrichment analysis. I used *Drosophila* phenotype controlled vocabulary terms from the FlyBase phenotype ontology [83]. The phenotypes resulting from single gene loci perturbations were used by DroPhEA and the same were used in the BiNGO analyses as background. About 4,800 genes had phenotypes associated with them. The p-values were corrected by Bonferroni correction in DroPhEA and by Benjamini and Hochberg FDR correction in BiNGO. The

enriched and depleted p-values obtained for gene lists were negative log transformed and scaled (0–1) to create a distance matrix and then clustered hierarchically. R heatmap.plus package was used to plot the scaled values. I used BiNGO to compare phenotype enrichment in variously filtered gene lists and also to compare phenotype enrichment between filtered gene lists and filtered networks. I used to DroPhEA to compare enrichment and depletion of phenotypes in different tissue and stage networks filtered for genes expressed at pmaxi >75 percent.

3.3 Results

3.3.1 Comparison of genes expressed ubiquitously or in specific tissues or developmental stages.

To examine the interaction properties of *Drosophila* genes expressed in different patterns I classified genes based on the specificity of their expression across tissues or developmental stages (Materials and Methods). I used tissue expression data from FlyAtlas [7], which covers 15 adult and 8 larval tissues, and developmental stage expression data from the modENCODE project [4], which covers 30 developmental times points from embryo to adult. I classified genes expressed predominantly in one tissue as tissue specific (2,838 genes), or in one stage as stage specific (3,566 genes). I classified genes expressed across all tissues or all stages as tissue ubiquitous (3,952 genes) or stage ubiquitous (4,969 genes), respectively. 3,226 of these genes are both tissue ubiquitous and stage ubiquitous; I refer to these as common ubiquitous genes (Table 3-1). Overall, I classified 24% of the *Drosophila* genes as ubiquitous and 19% as tissue-specific in this study. This is comparable to a human study [151] in which 26% of the genes were

classified as ubiquitous across 15 human tissues and cell lines and 13% were considered tissue-specific. As expected, the *Drosophila* ubiquitously expressed genes are enriched for genes involved in basic cellular processes, including protein synthesis, trafficking, and degradation, RNA transcription and processing, and cytoskeleton organization. The ubiquitously expressed genes are also enriched for intracellular proteins while the tissue specific or stage specific genes are enriched for extracellular proteins (Table 3-2). This is in partial agreement with the results of a human study where the tissue-specific proteins were found to be enriched for extracellular and membrane proteins [151]. The *Drosophila* ubiquitously expressed genes are also more evolutionarily conserved than the tissue or stage specific genes (Figure 3-1). For example, about 37% of the ubiquitously expressed genes have yeast orthologs while only 12% of the tissue-specific genes and 8% of the stage-specific genes have yeast orthologs. Clear human orthologs exist for 80% of the ubiquitously expressed genes and only 19% and 14% of the tissue and stage specific genes, respectively. Among the genes that have orthologs in yeast or metazoans, about 50% are ubiquitously expressed while only 9–12% are tissue-specific and only 11–15% are stage-specific. This is in agreement with several studies examining the conservation of ubiquitously expressed and tissue-specific proteins [162-166].

3.3.2 Tissue and stage specific proteins predominantly interact with ubiquitously expressed proteins and not with each other.

To determine the interaction properties of the *Drosophila* proteins encoded by genes that are expressed ubiquitously or in specific stages or tissues I examined the 235,950 protein-protein interactions (PPI) available in the DroID database [98]. These

Table 3-2. GO term cellular component enrichment for the ubiquitous and specific genes. Top six enriched GO cellular component (CC) terms for the specific genes and top ten enriched GO CC terms for the ubiquitous genes are shown.

GO Term Enrichment for Stage Specific Genes

GO Term - Cellular Component	p-value
GO:0005576~extracellular region	1.04E-64
GO:0005615~extracellular space	8.94E-20
GO:0044421~extracellular region part	1.92E-12
GO:0042600~chorion	4.91E-10
GO:0030312~external encapsulating structure	1.99E-09
GO:0016222~procollagen-proline 4-dioxygenase complex	2.61E-09

GO Term Enrichment for Tissue Specific Genes

GO Term - Cellular Component	p-value
GO:0005576~extracellular region	3.66E-13
GO:0030286~dynein complex	2.79E-10
GO:0030312~external encapsulating structure	8.09E-10
GO:0042600~chorion	4.34E-09
GO:0016222~procollagen-proline 4-dioxygenase complex	2.29E-08
GO:0015630~microtubule cytoskeleton	2.05E-06

GO Term Enrichment for Stage Ubiquitous Genes

GO Term - Cellular Component	p-value
GO:0005622~intracellular	1.43E-117
GO:0044424~intracellular part	8.63E-79
GO:0005623~cell	4.69E-72
GO:0044464~cell part	4.69E-72
GO:0043227~membrane-bounded organelle	2.25E-63
GO:0043231~intracellular membrane-bounded organelle	4.22E-63
GO:0043226~organelle	1.75E-59
GO:0043229~intracellular organelle	2.38E-59
GO:0044422~organelle part	3.24E-49
GO:0044446~intracellular organelle part	3.47E-49

GO Term Enrichment for Tissue Ubiquitous Genes

GO Term - Cellular Component	p-value
GO:0005622~intracellular	2.23E-86
GO:0044424~intracellular part	2.47E-71
GO:0005737~cytoplasm	3.26E-62
GO:0032991~macromolecular complex	1.56E-52
GO:0005623~cell	1.14E-51
GO:0044464~cell part	1.14E-51
GO:0044422~organelle part	1.15E-50
GO:0044446~intracellular organelle part	3.31E-50
GO:0043226~organelle	9.36E-44
GO:0043229~intracellular organelle	3.18E-43

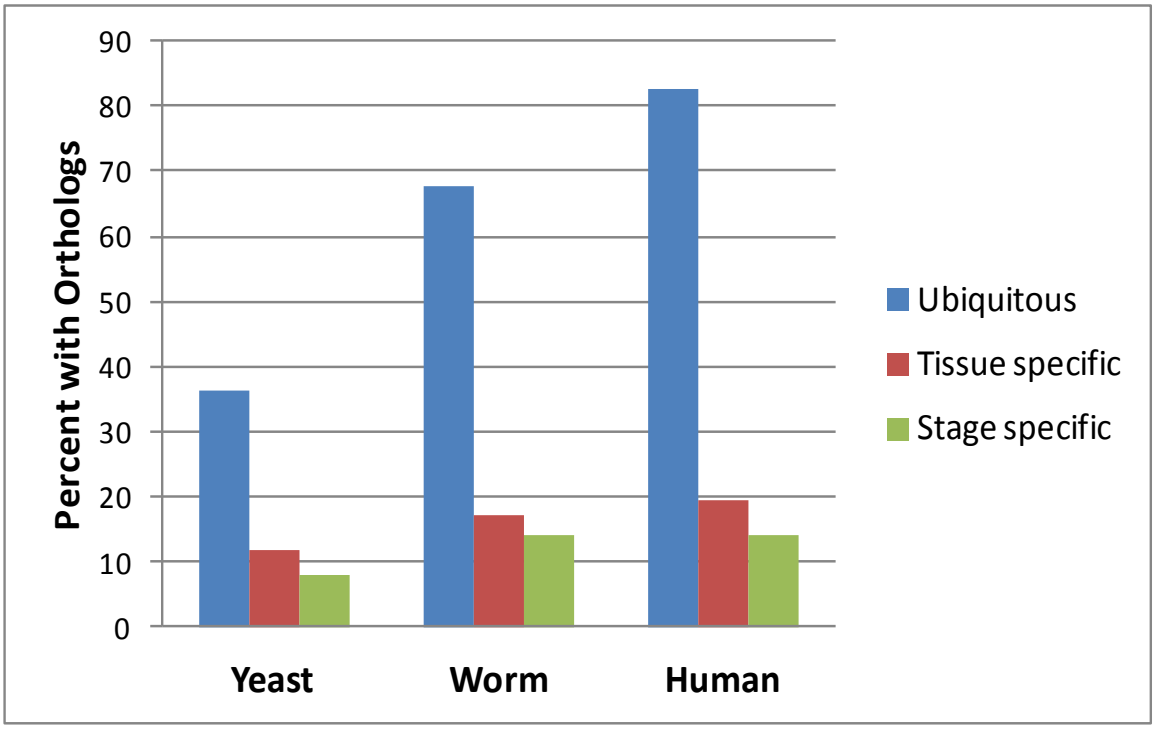


Figure 3-1. Conservation of ubiquitously expressed genes. Bars indicate the percentage of ubiquitous, tissue-specific, and stage-specific *Drosophila* genes that are conserved in each organism. Conservation is based on identification of close sequence homology (Methods).

include 93,544 interactions that were detected experimentally with *Drosophila* proteins, and 144,171 interactions predicted from experimental data in other species (interologs). For simplicity I refer to proteins encoded by ubiquitously expressed genes as "ubiquitous proteins", and likewise to "stage-specific" and "tissue-specific proteins", keeping in mind that measured mRNA abundance may be an imperfect surrogate for protein abundance [167]. I found that the ubiquitous proteins have about three times more interactions per protein than the stage-specific or tissue-specific proteins (Table 3-3). When interologs are removed from the analysis (Table 3-3, numbers in parentheses), the ubiquitous proteins still have about three times more interactions than the tissue or stage specific proteins, indicating that the difference is not due simply to the fact that ubiquitous proteins are more conserved and therefore have more interactions that are interologs. This finding is in general agreement with findings for an analysis of human proteins in which proteins expressed in more tissues or cell lines generally had more interactions [151]. Next I looked at the numbers of interactions within and between the ubiquitous and the specific proteins. Roughly half of the protein interactions that involve ubiquitous proteins are with other ubiquitous proteins while only 12 – 15% of their interactions are with the specific proteins (Table 3-3). In contrast, fewer than 10% of the interactions involving the stage-specific or tissue-specific proteins are with other specifically expressed proteins, whereas around 60% or their interactions are with ubiquitous proteins (Table 3-3). Again, these differences were found independent of whether or not the interolog data were included. The finding that *Drosophila* tissue-specific proteins tend to interact with a core set of ubiquitously expressed proteins is consistent with analyses of the human interactome

Table 3-3 Interactions involving ubiquitously and specifically expressed proteins. Numbers are for all *Drosophila* protein interactions including those predicted from other species (interologs) but not measured with *Drosophila* proteins. Numbers in parentheses are for protein interactions detected experimentally with *Drosophila* proteins.

			Percent of interactions with:		
	Proteins	Interactions/ protein	Ubiq.	NUNS	Specific
Tissue-ubiquitous	3,960	37 (15)	49 (41)	37 (43)	14 (16)
Stage-ubiquitous	4,972	35 (14)	48 (46)	40 (37)	12 (16)
Tissue-specific	2,838	12 (6)	59 (54)	33 (40)	8 (7)
Stage-specific	3,656	10 (5)	56 (63)	35 (29)	9 (8)

[150, 151]. Combined, these results suggest that ubiquitous proteins tend to interact with each other while tissue or stage specific proteins tend to interact with ubiquitous proteins and that this is a conserved feature of protein interaction networks.

The above analyses relied on counting interactions within and between groups of proteins, which may not be a good measure of the interaction tendencies of those groups. For example, if a group of proteins has few overall interactions (as the specific proteins do), then they will likely have few interactions among themselves just by chance. Likewise, proteins with many overall interactions are more likely to interact with each other just by chance. Moreover, since the protein interaction networks generally follow a power law distribution of interactions per protein (degree) [17], the average degree for a group of proteins may not be representative or useful for comparing between groups. Thus, to further compare the intrinsic interaction tendencies of the ubiquitously and specifically expressed *Drosophila* proteins I calculated fold difference between the number of interactions among or between these groups and random sets of proteins of the same sizes (Methods). I found that both the stage and tissue ubiquitous proteins had about 8-fold (p-value 7.6×10^{-5}) more interactions among themselves compared to random networks (Figure 3-2). In contrast the tissue and stage specific PPI networks had about 5–6 fold (p-value 7.6×10^{-5}) fewer interactions than random (Figure 3-2).

It is perhaps not surprising to find few interactions among all of the stage or tissue specific proteins, since many of them are never expressed together in the same tissue or at the same developmental stage. Thus I set out to determine whether the stage or tissue specific proteins interacted with each other within each stage or tissue. To do this, I built

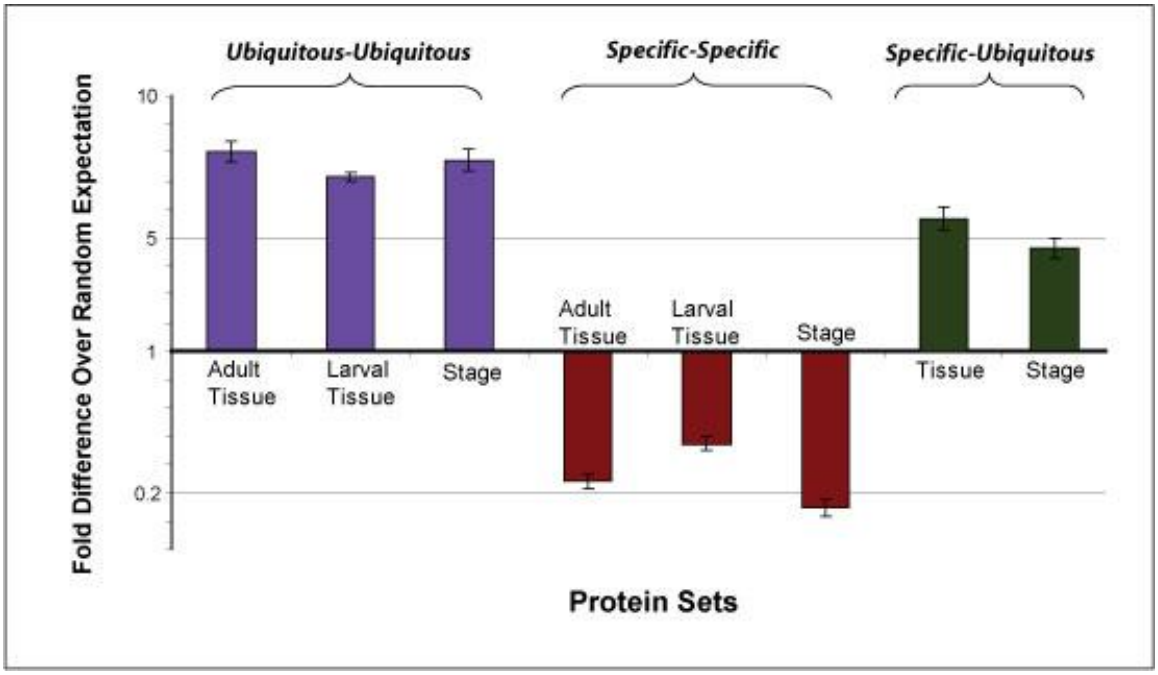
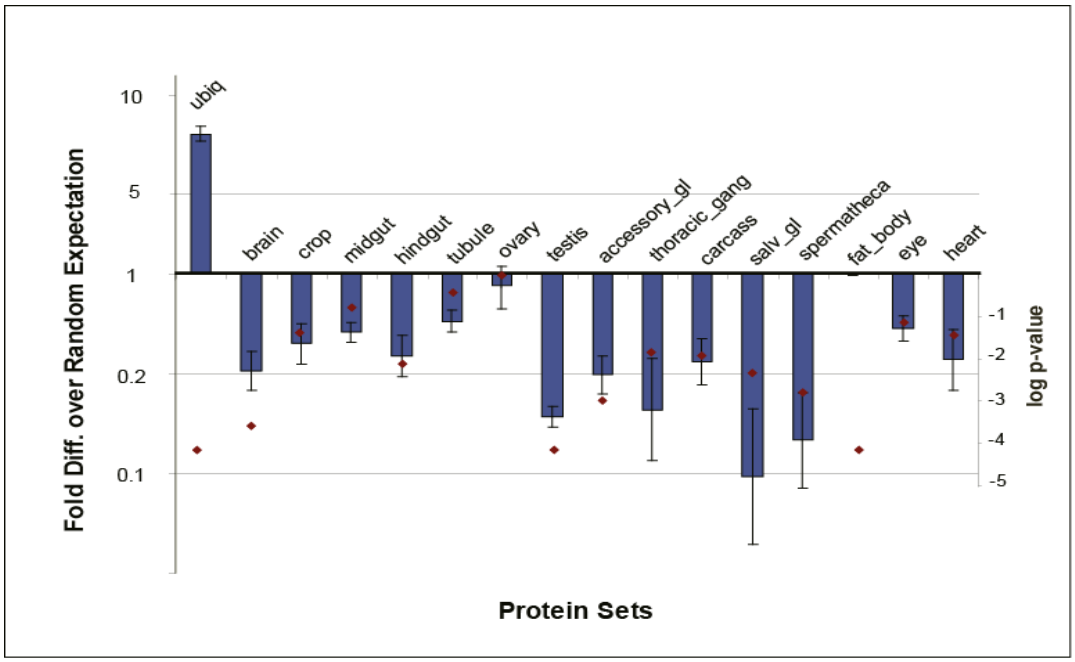


Figure 3-2 Ubiquitous proteins frequently interact with each other while tissue and stage specific proteins tend to interact with ubiquitous proteins but not each other. For each set of proteins the number of interactions within the set (ubiquitous-ubiquitous and specific-specific) or between sets (ubiquitous-specific) was compared to the number of interactions in each of 5,000 random protein sets taken from the composite PPI network. Each bar shows average fold-difference for 5,000 trials and standard deviation. The p-value significance for each case is <7.56e-5 at a CI = 99.99%. Ubiquitous-ubiquitous interactions were tested with adult tissue ubiquitous proteins, larval tissue ubiquitous proteins, and stage ubiquitous proteins. Specific-specific interactions were tested with adult tissue specific proteins, larval tissue specific proteins, and stage specific proteins. Ubiquitous-specific interactions were tested between adult tissue ubiquitous and tissue specific proteins or between stage ubiquitous and stage specific proteins.

tissue-relevant networks for each of the 23 adult and larval tissues and performed the same comparisons. The tissue-specific proteins within every tissue except the ovary had many fold fewer interactions among themselves than did random sets of proteins (Figure 3-3A). This is in agreement with a human study which found that interactions between tissue-specific proteins in different human tissues was rare [151]. Similarly, stage-specific proteins at each of the 30 developmental time points were depleted for interactions among themselves, with the exception of proteins specifically expressed in the one-day old female and the 6-8 hour embryo (Figure 3-3B). I also found that the tissue specific or stage specific proteins rarely interacted among themselves indirectly through third non-specific proteins either in the tissue or stage specific networks (Figure 3-4). I did the same analyses with larval tissue networks and obtained similar results (Figure 3-5). Thus, tissue and stage specific proteins generally do not form well-connected subnetworks where they are expressed. As the tissue or stage specific proteins rarely interacted with each other directly or indirectly, I determined whether they primarily interacted with the ubiquitous proteins. Both the tissue specific and stage specific proteins interacted about five-fold more with the ubiquitous network than did random sets of proteins (Figure 3-2). These findings were largely independent of the methods used to detect the protein interactions. When I did separate analyses with interactions detected by yeast two hybrid and those detected mostly by co-complex studies, I found that for both methods the ubiquitously expressed proteins tend to interact with each other, the tissue or stage specific proteins tend not to interact with each other, and the specifically expressed proteins tend to interact with the ubiquitous proteins

A.



B.

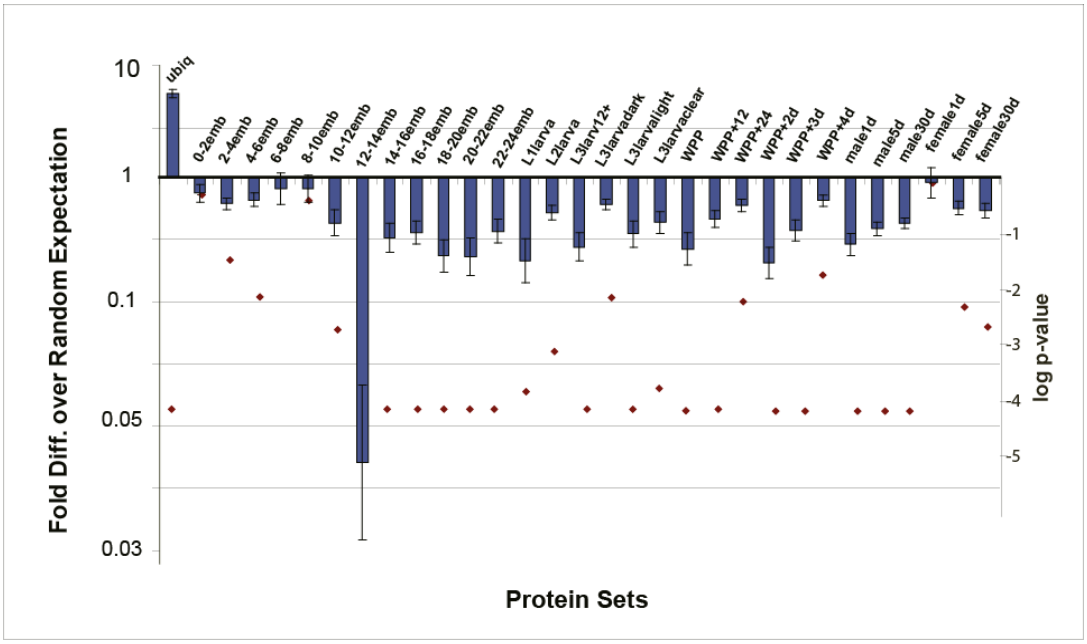
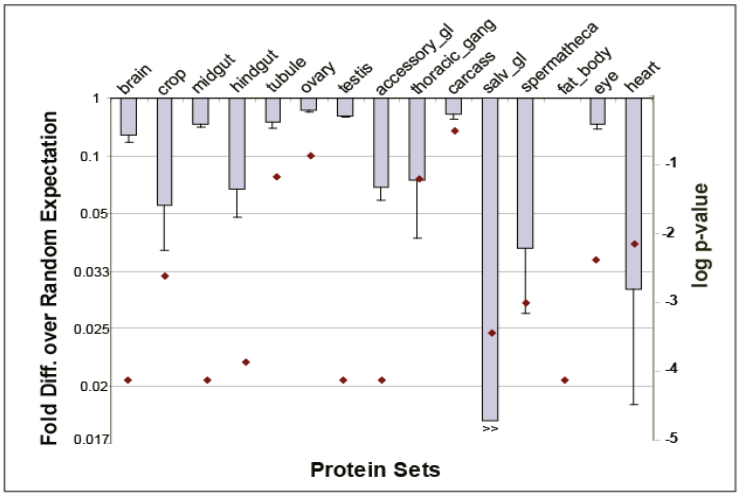


Figure 3-3 Within tissue or stage specific subnetworks, tissue or stage specific proteins tend not to interact with each other. For each protein set the average fold

difference between the number of direct interactions in the test set and the number of interactions among proteins in each of 5,000 random protein sets is shown. Standard deviations are shown as error bars. The p-values for each comparison are shown as red dots (log p-value on the right axis). **A.** Interactions among the tissue ubiquitous proteins or among each set of tissue specific proteins in each of 15 adult tissues. **B.** Interactions among the stage ubiquitous proteins or among each set of stage specific proteins for each of 30 developmental stages.

A.



B.

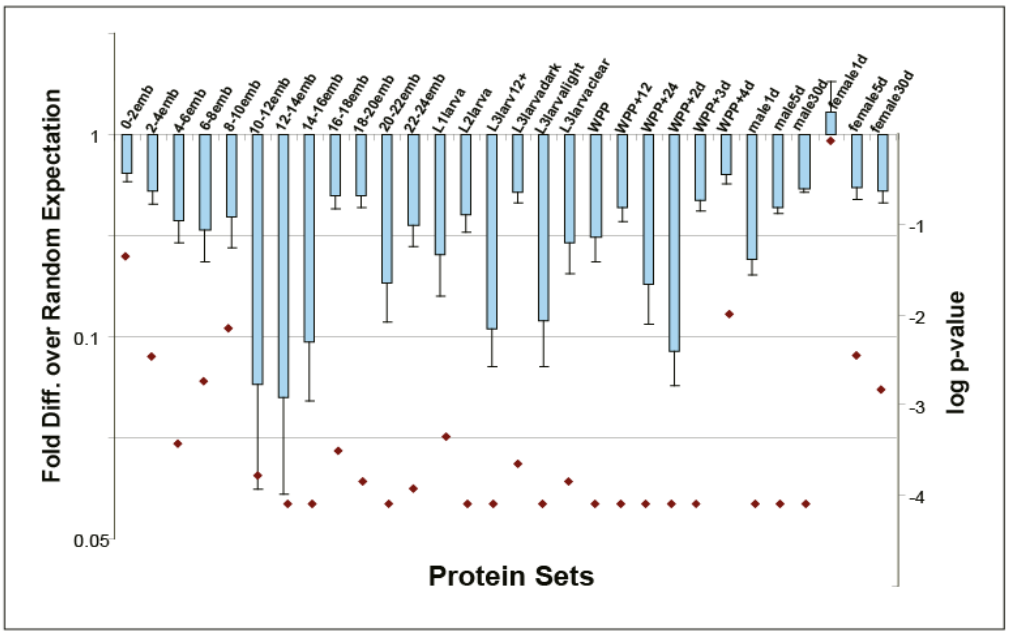
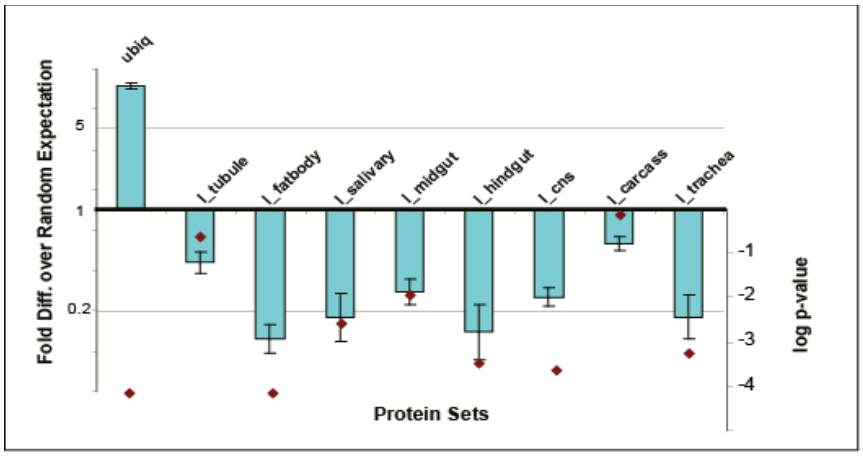


Figure 3-4 Within tissue or stage specific subnetworks, tissue or stage specific proteins tend not to interact with each other indirectly through a third protein.

For each protein set the average fold difference between the number of indirect interactions in the test set and number of indirect interactions among proteins in each of

5,000 random protein sets is shown. Standard deviations are shown as error bars. The p-values for each comparison are shown as red dots (log p-value on the right axis). **A.** Indirect interactions among the tissue ubiquitous proteins or among each set of tissue specific proteins in each of 15 adult tissues. **B.** Indirect interactions among the stage ubiquitous proteins or among each set of stage specific proteins for each of 30 developmental stages.

A.



B.

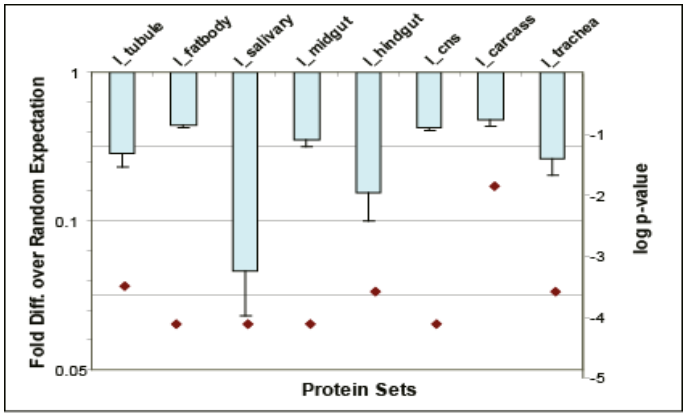
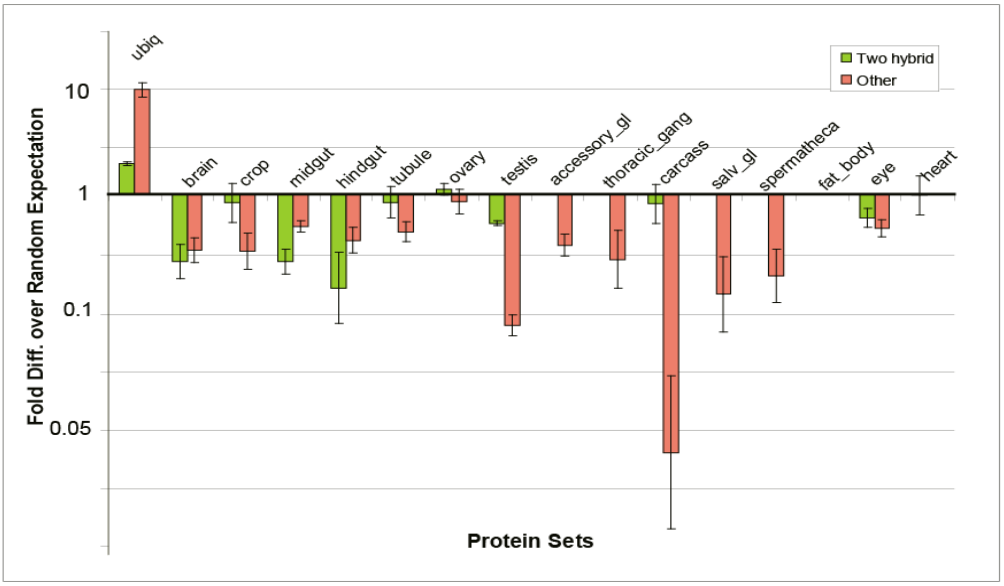


Figure 3-5 Within larval tissue subnetworks, tissue specific proteins tend not to interact with each other directly or indirectly. For each protein set the average fold difference between the number of direct (A) or indirect (B) interactions in the test set and number of direct or indirect interactions among proteins in each of 5,000 random protein sets is shown. Standard deviations are shown as error bars. The p-values for each comparison are shown as red dots (log p-value on the right axis). **A.** Direct interactions among the larval tissue ubiquitous proteins or among each set of larval tissue specific proteins. **B.** Indirect interactions among each set of larval tissue specific proteins.

A.



B.

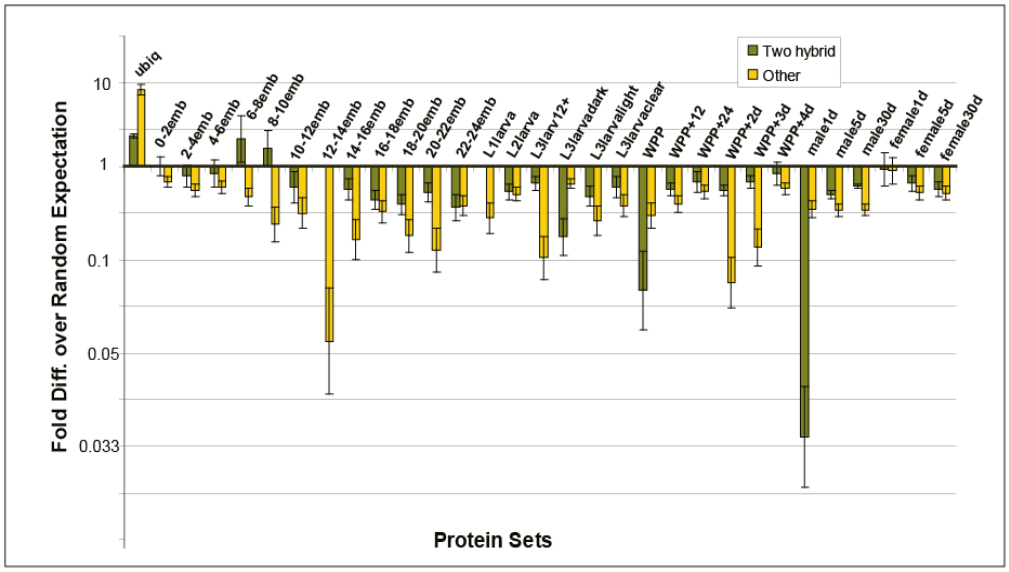


Figure 3-6 Interaction properties specifically expressed and ubiquitous proteins are largely method independent. For each protein set the average fold difference between the number of direct interactions in the test set and the number of interactions among proteins in each of 5,000 random protein sets is shown. Standard deviations are shown as

error bars. I divided the interaction data into a set detected solely by yeast two hybrid (44,880 interactions among 9,335 proteins) and a set not detected by yeast two hybrid (201,395 interactions among 8,141 proteins), at least 67% of which were detected by complex pull down. **A.** Interactions among the tissue ubiquitous proteins or among each set of tissue specific proteins in each of 15 adult tissues. **B.** Interactions among the stage ubiquitous proteins or among each set of stage specific proteins for each of 30 developmental stages.

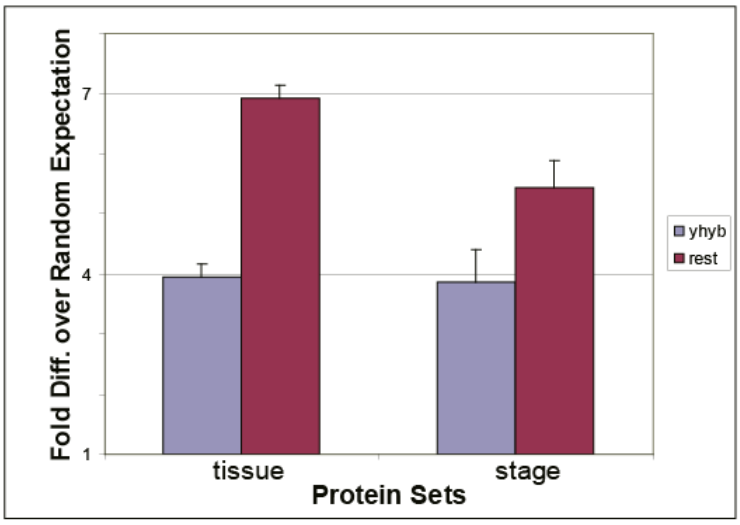


Figure 3-7 Tissue and stage specific proteins interact with the ubiquitous proteins irrespective of method used. For all adult tissue specific proteins or all the stage specific proteins the average fold difference of the number of their direct interactions with ubiquitous proteins over direct interactions of each of 5,000 random proteins with the ubiquitous proteins is shown. The standard deviations of fold are shown as error bars. The analysis was done separately with the protein interactions detected by yeast two hybrid and those detected by other methods.

(Figure 3-6 and 3-7). From these combined results I can conclude that the tissue or stage specific proteins primarily interact with the ubiquitous core network, most likely to modify it for tissue or stage-specific functions.

3.3.3 Identification of subnetworks that are active in specific contexts

Because the techniques used to collect much of the available interactome data are context independent, filters are needed to identify the subnetworks that operate in specific tissues or at specific developmental times. One type of filter that can be envisioned is one that retains only genes that are specifically expressed in a tissue or stage. However, as demonstrated above for *Drosophila* and elsewhere for human [150-152], proteins expressed in specific tissues or stages tend to interact with ubiquitously expressed proteins rather than with other specifically expressed proteins. In the PDI, ubiquitous and non-ubiquitously expressed genes might be regulated by both types of TFs. Thus, a PPI or PDI network filter based on expression specificity is likely to remove many interactions that are relevant in specific contexts. A second type of filter is one that relies on absolute expression levels. For example, frequently genes are classified as "on" or "off" in a particular sample based on arbitrary expression thresholds. Alternatively, genes can be ranked based on their relative expression levels within a particular tissue and such rankings can be used to identify the genes most highly expressed in a particular context. The problem with expression level filters, however, is that different genes may be functional at widely different expression levels; two genes may differ in their expression levels by several orders of magnitude even when they expressed at their maximal level. I reasoned that a gene is more likely to be active when it is expressed at or near its

maximal level relative to its level in all tissues or stages. For example, if a gene is expressed across all tissues but is maximally expressed in the ovary, it is likely to have a function in the ovary. To test this hypothesis I developed a scale to indicate the fraction of maximal expression for each gene in each tissue or developmental stage. For each tissue or stage I calculate a gene's expression level as a percent of its maximal level across all tissues or stages. The percent maximum or "pmax" value is a gene's expression value normalized to its maximum expression. To evaluate the pmax scale, I first asked if I could obtain gene lists enriched for tissue-relevant functions by selecting genes expressed above different pmax thresholds (e.g., pmax >50 or >75) in specific tissues. I compared these filters to one that selects genes expressed above the average value for all genes in a given tissue. I chose a set of six tissues (brain, thoracic ganglion, eye, testis, ovary and larval CNS) and applied the three different filters to genes expressed in those tissues (Table 3-4). The expression filters were evaluated by checking for enrichment of tissue-relevant mutant phenotypes in the respective gene lists. The results show that the pmax filtered gene lists are highly enriched for tissue-relevant phenotypes in contrast to the gene lists filtered on average expression, which show little or no enrichment for tissue-relevant phenotypes (Figure 3-8). The >75 pmax filter seems to perform better than the >50 pmax for the selected tissues, supporting the hypothesis that genes expressed closer to their maximum level in a tissue are more likely to be functional in that tissue.

To test whether pmax values could be used to identify biologically relevant subnetworks in the PPI interactome I selected subnetworks containing interactions between genes expressed above 75 pmax in specific tissues and computed phenotype

Table 3-4 Number of genes in the tissue gene lists after applying different expression filters. Number of genes expressed at average, greater than 50 pmax and greater than 75 pmax in six different tissues.

gene list	# brain	# ovary	# testis	# thoracic gang	# larval CNS	# eye
average	2,020	1,907	2,119	1,732	2,211	1,462
pmax 50	2,754	3,469	2,704	2,437	2,984	1,598
pmax 75	1,525	2,502	2,277	1,087	1,532	703
total expressed	7,527	6,452	8,148	7,124	7,598	7,222

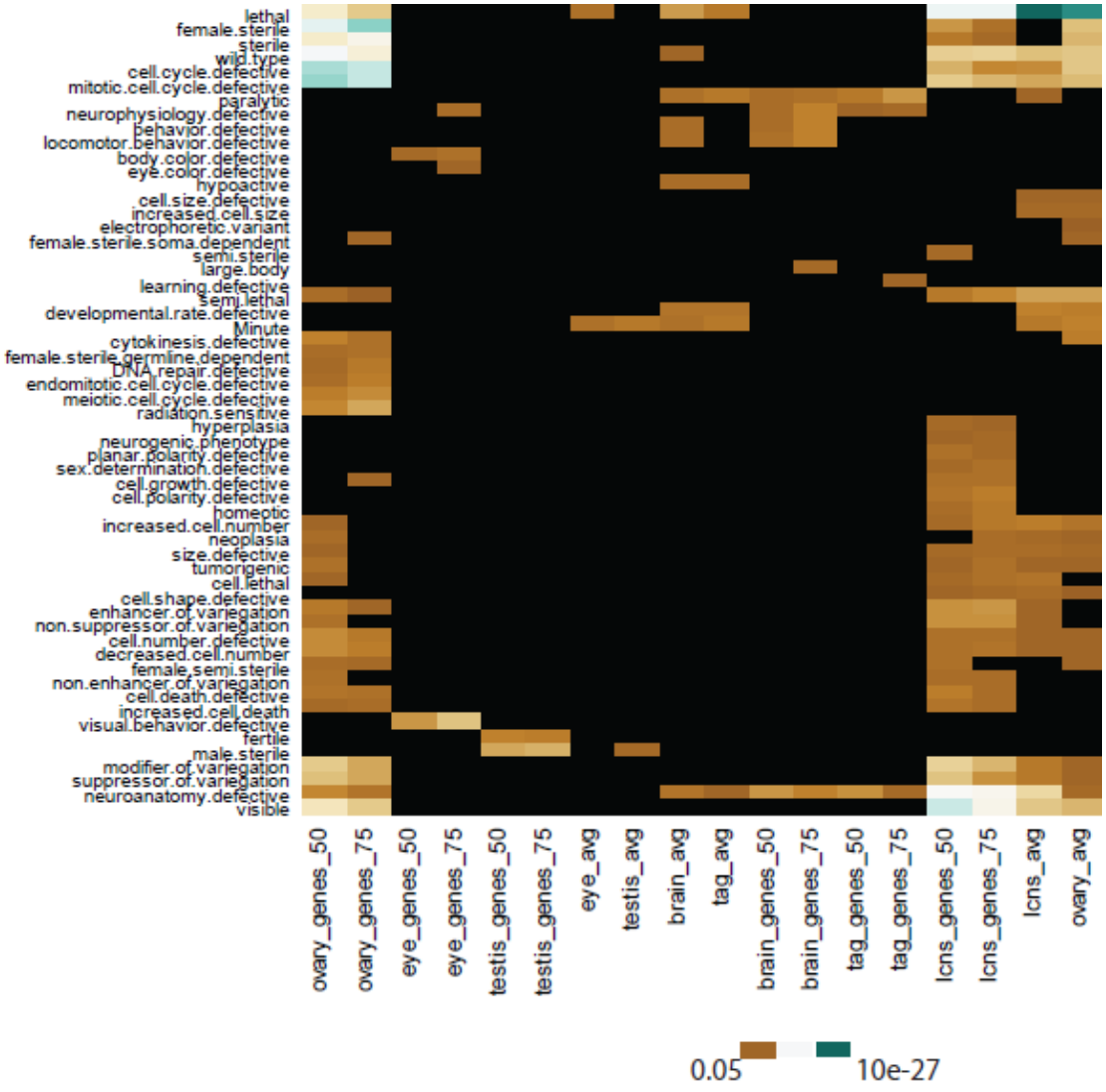


Figure 3-8 Heat map of enriched phenotypes in variously filtered gene lists. Heat map of enriched phenotypes obtained when genes expressed above the average, greater than 50 pmax and greater than 75 pmax in six different tissues were used as query genes and compared to background. The p-values were log transformed, scaled and then plotted.

enrichment and depletion. The PPI subnetworks of genes expressed at 75 pmax in specific tissues were highly enriched for tissue-relevant phenotypes (Figure 3-9 and 3-10). For example, the ovary subnetwork is enriched for female sterile phenotype while the testis subnetwork is enriched for male sterile phenotype (Figure 3-9). The brain and thoracic ganglion networks are enriched for neuroanatomy and neurophysiology phenotypes while the eye subnetwork is enriched for visual behavior phenotypes. Likewise, many of the PPI subnetworks expressed at 75 pmax in specific stages were enriched for stage relevant phenotypes (Figure 3-11). The early embryo networks, for example, were enriched for genes annotated with cell cycle phenotypes as expected given that early embryogenesis is dominated by cell division [168]. As development progresses from early embryo to late embryo, the subnetworks become enriched for behavioral phenotypes. As development progresses into larval and pupal stages and through to adult, the subnetworks show little phenotype enrichment. This perhaps reflects the differentiation of postembryonic organs into collections of diverse tissues, where it would be more appropriate to use expression data in relevant tissues to identify context-relevant subnetworks. The tissue and stage 75 pmax subnetworks are also depleted for phenotypes that would not be expected for genes in specific tissues or stages (Figures 3-12 and 3-13). For example, the subnetworks for ovary and early embryo stages are depleted of genes involved in behavioral phenotypes.

The ability of pmax filters to identify context relevant subnetworks is also supported by similarity of subnetworks based on the phenotype terms enriched in them. In the case of the tissue-relevant pmax PPI subnetworks (Figure 3-10), the ovary and

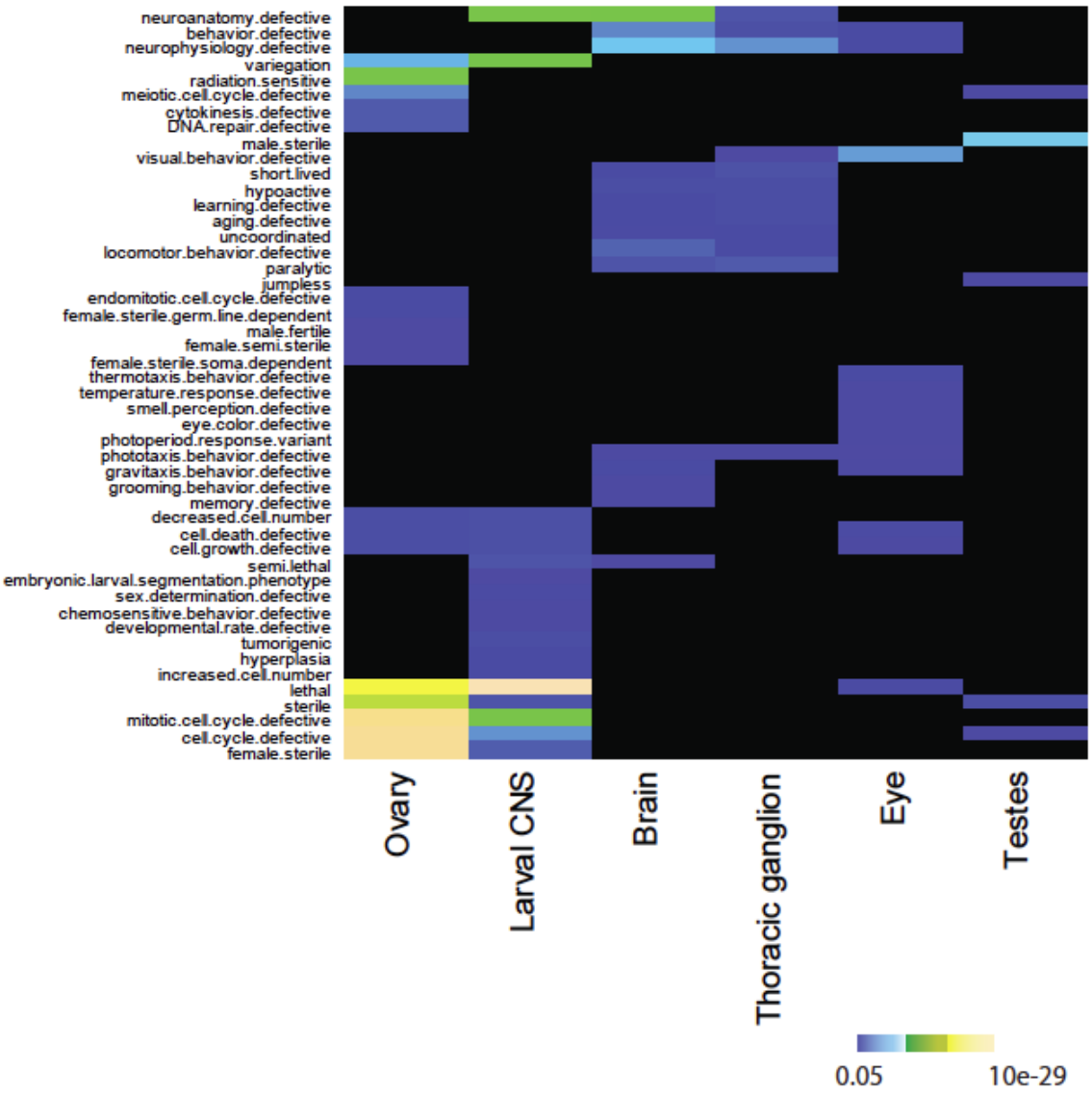


Figure 3-9 Heat map of phenotypes enriched in six tissue-relevant subnetworks. Heat map of enriched phenotypes obtained when the nodes of >75 percent of maximum subnetworks for ovary, larval CNS, brain, thoracic ganglion, testis and eye were included as query genes and compared to the background. The raw p-values were log transformed, scaled and then plotted.

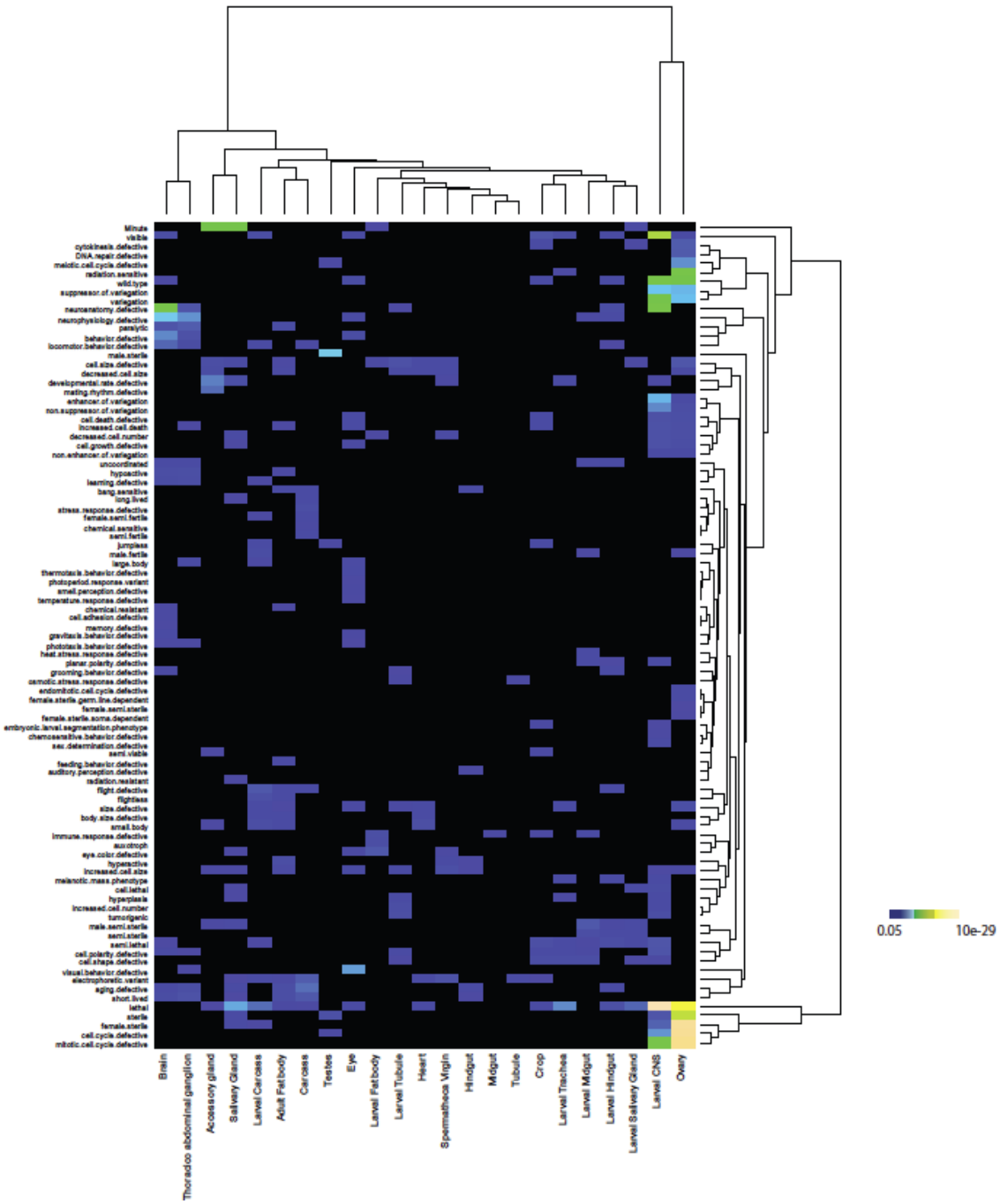


Figure 3-10 Heat map of phenotypes enriched in tissue-relevant subnetworks. Heat map of enriched phenotypes obtained when the nodes of >75 percent of maximum subnetworks for each tissue were included as query genes and compared to the background. The raw p-values were log transformed, scaled and then plotted.

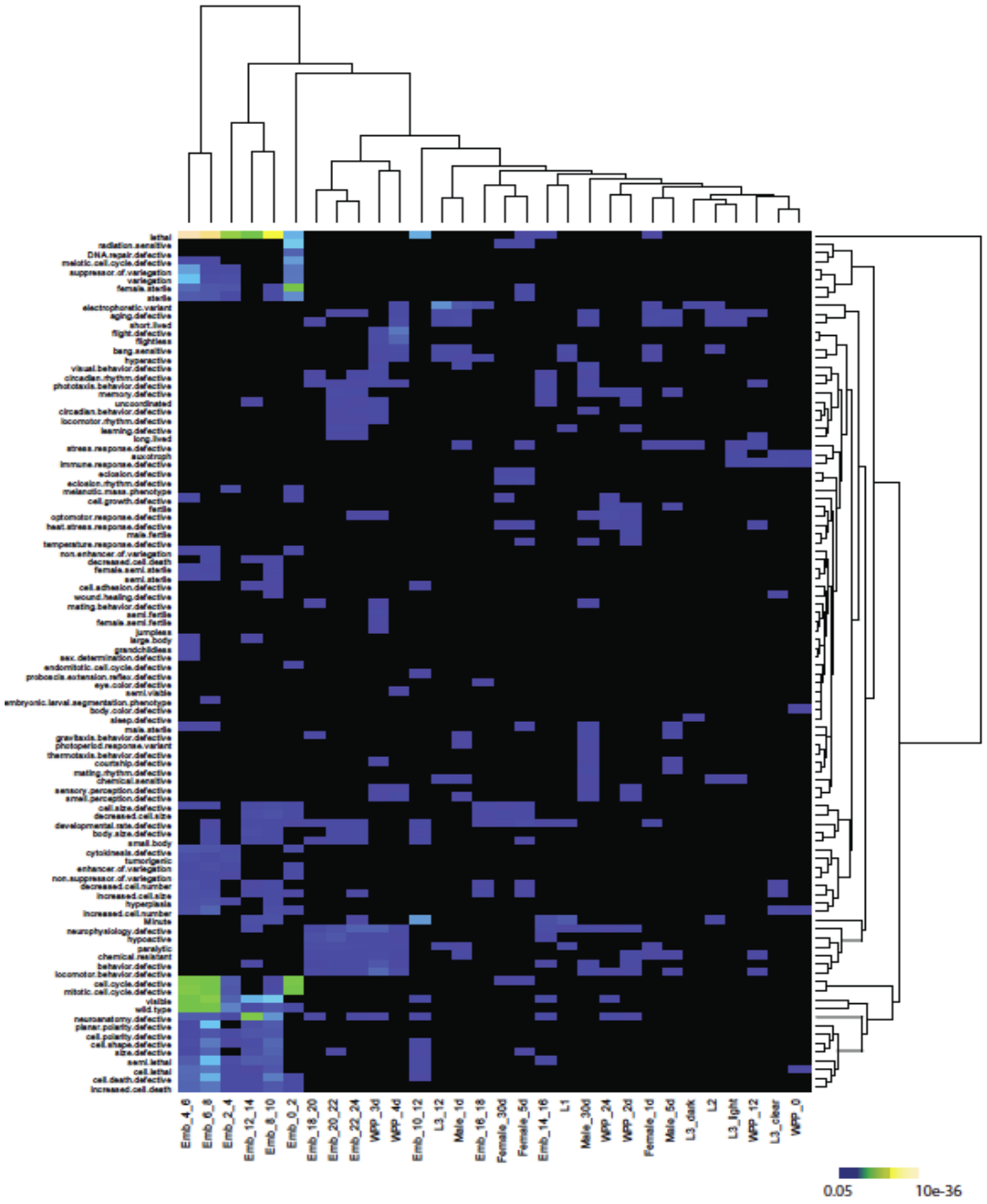


Figure 3-11 Heat map of phenotypes enriched in stage-relevant subnetworks. Heat map of enriched phenotypes obtained when the nodes of >75 pmax subnetworks for each stage were included as query genes and compared to the background. The raw p-values were log transformed, scaled and then plotted.

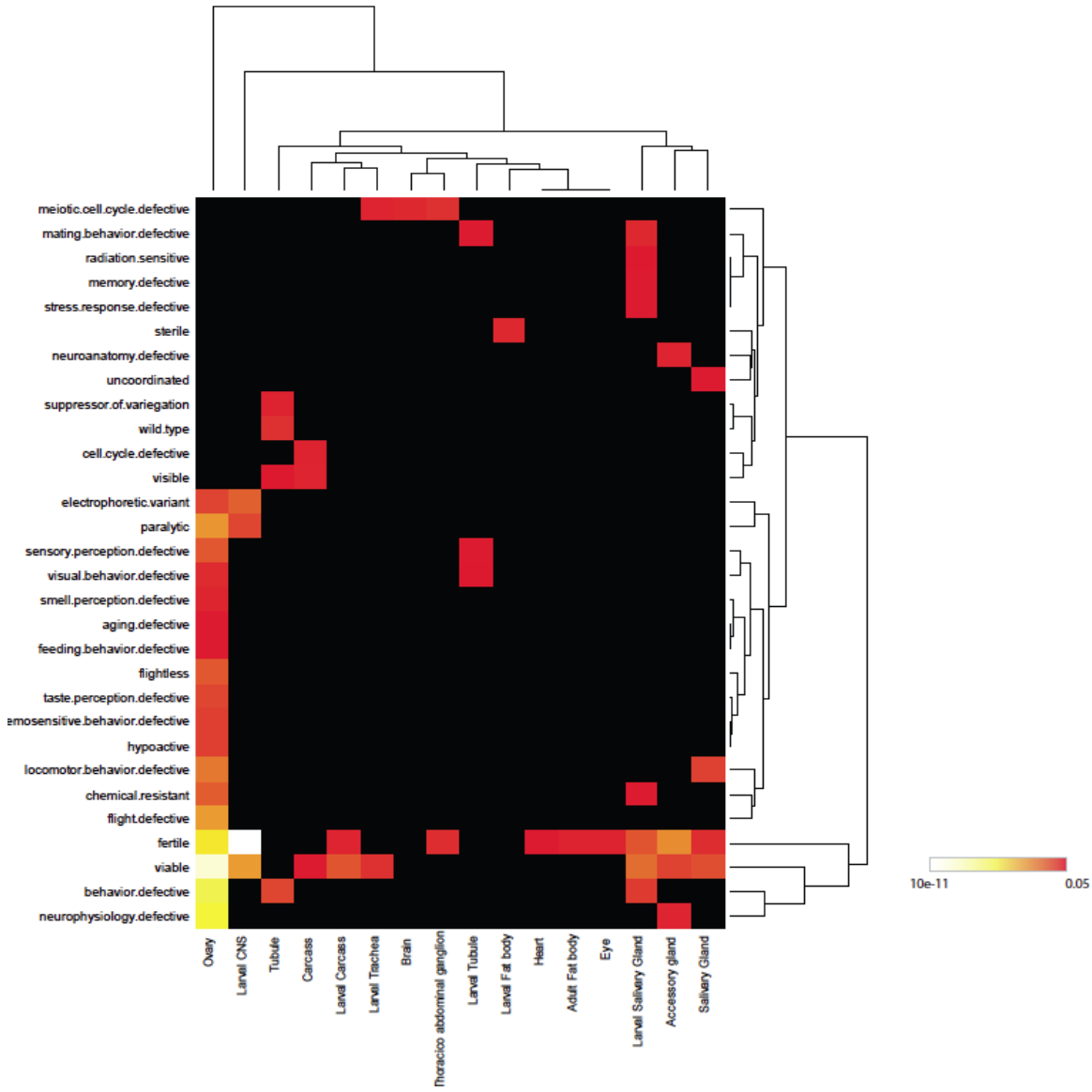


Figure 3-12 Heat map of depleted phenotypes in tissue-relevant subnetworks. Heat map of depleted phenotypes obtained when nodes of the >75 pmax subnetworks for each tissue were used as the query genes and compared to the background. The raw p-values were log transformed, scaled and then plotted.

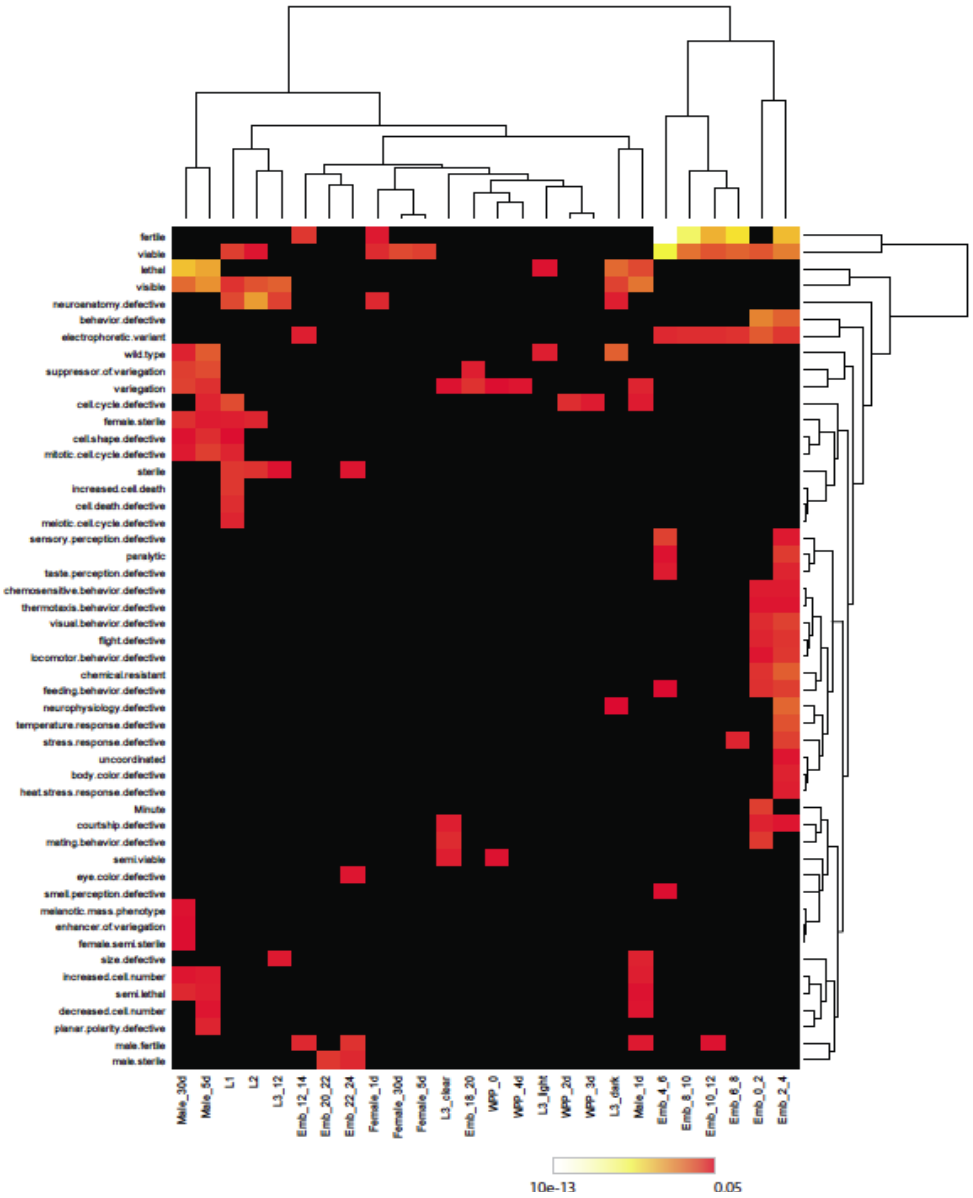


Figure 3-13 Heat map of depleted phenotypes in stage-relevant subnetworks. Heat map of depleted phenotypes obtained when nodes of the >75 pmax subnetworks for each stage were used as the query genes and compared to the background. The raw p-values were log transformed, scaled and then plotted.

larval CNS cluster together as they are both enriched in genes that share 'cell cycle' and 'lethality' phenotypes among others. This is perhaps not surprising as it is has been shown that many cell cycle and maternal genes play major roles in asymmetric cell divisions in nervous system development [169, 170]. Another example is brain and thoracic ganglion pmax PPI subnetworks clustering together based on shared neuroanatomy and neurophysiology phenotypes among others. The testis subnetwork is an outgroup as it alone is enriched for 'male sterile' phenotype, and similarly, the eye subnetwork is the only one enriched for visual perception related phenotypes. Most larval tissue subnetworks cluster together due to their enrichment for lethality phenotypes. The early embryo subnetworks also cluster together, as do the late embryo subnetworks along with late pupal subnetworks (Figure 3-11). This might reflect differentiation events taking place both in late embryos and late pupae in contrast to the intervening larval stages where primarily growth takes place [171]. The subnetwork at the mid-embryo stage from 10–12 hr forms an outgroup, consistent with studies [146, 172] showing that early-embryo specific transcripts are down-regulated and the late embryo-specific transcripts are just beginning to be expressed. The embryo subnetworks at 14–16 hr and 16–18 hr also do not cluster with the other embryo subnetworks and are not significantly enriched for embryo relevant phenotypes, potentially signifying a transition to the late embryo stages. Another subnetwork that stands out is from the 0–2 hr embryo, which is unique in that only rapid nuclear divisions are taking place. Consistent with its requirement for rapid cycles of mitosis and DNA synthesis, the 0-2 hour embryo subnetwork is enriched for phenotypes such as 'radiation sensitive', 'DNA repair defective' and 'cell cycle

defective'. It has been shown that related tissues show similar expression profiles and that tissues in consecutive developmental stages cluster together based on their gene expression patterns [173]. Here I show that related tissue maximum PPI subnetworks and as well as consecutive stage maximum expressed PPI subnetworks cluster together based on shared phenotypes.

I compared the phenotype enrichment of the 75 pmax subnetworks with similarly filtered gene lists. For this I picked the maximally expressed subnetworks and gene lists of the ovary, larval CNS, brain and thoracic ganglion (Figure 3-14). For example, applying a gene list filter to the ovary results in 2,502 genes expressed at >75 pmax expression level, while applying the >75 pmax filter to identify the ovary subnetwork results in a subnetwork with 2,085 of these genes; to be maintained in the filtered subnetwork a gene at >75 pmax must interact with another genes at >75 pmax. As shown in Figure 3-14, the ovary subnetwork is more enriched than the ovary gene list for ovary-relevant phenotypes such as 'female sterile', 'mitotic cell cycle defective', and 'cell cycle defective'. Similar results were obtained with the other tissues (Figure 3-14) and with the PDI network (data not shown). Thus, pmax filtered subnetworks are better enriched for tissue relevant phenotypes compared to the gene lists filtered at the same pmax. This is likely due to the fact that genes with related functions are frequently connected in the PPI and PDI subnetworks while genes with unrelated functions are more likely to be unconnected. Overall, these results suggest that the pmax filter is a useful method to identify the PPI and PDI subnetworks that operate in specific tissue contexts.

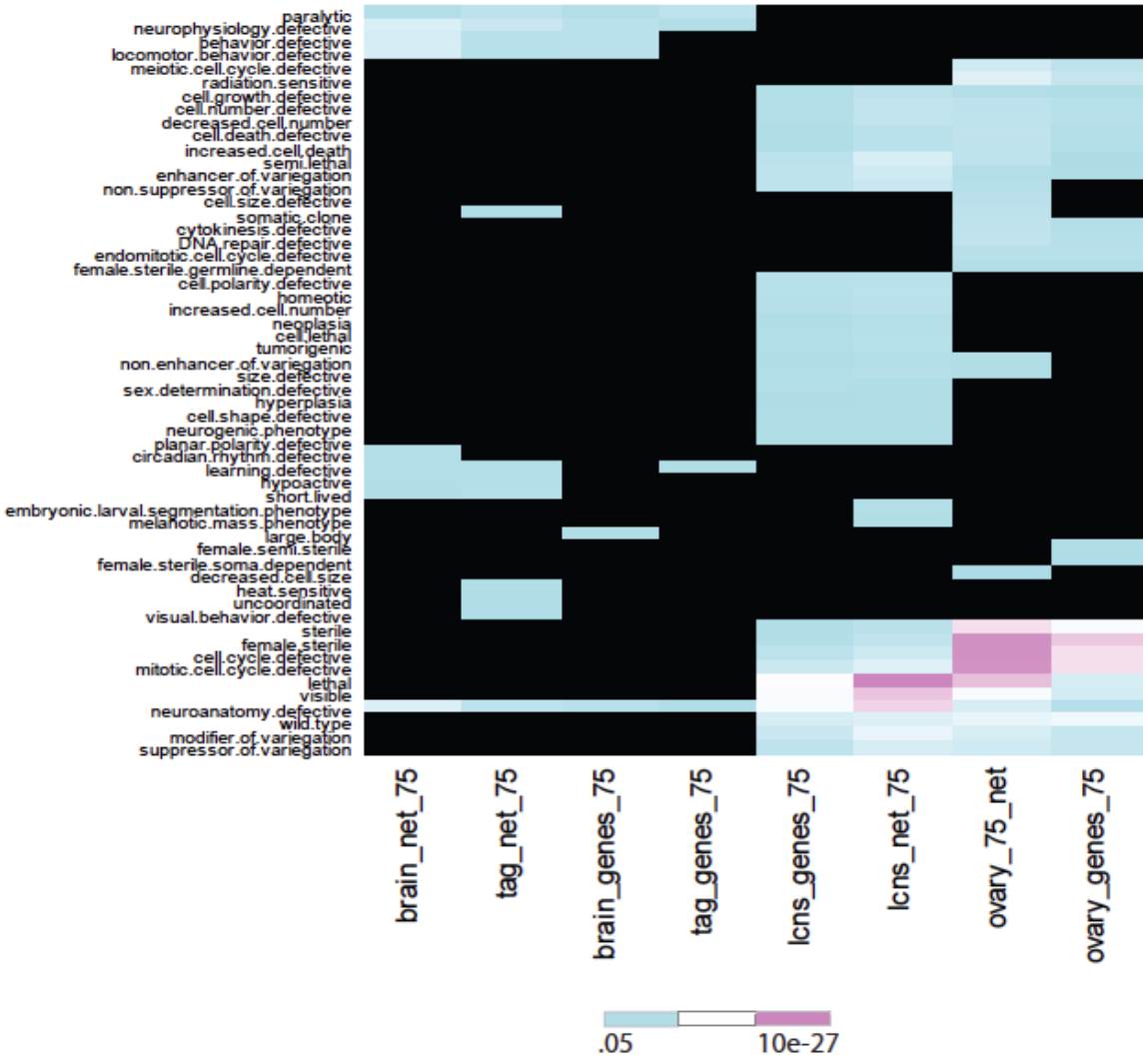


Figure 3-14 Heat map of enriched phenotypes in 75 pmax filtered networks compared to 75 pmax filtered gene lists. Heat map of enriched phenotypes obtained when genes expressed at 75 pmax and networks of genes expressed at 75 pmax in 4 different tissues were used as query genes and compared to background. The p-values were log transformed, scaled and plotted.

3.3.4 Examples of network modules identified with the pmax expression filter

Next I wanted to find examples of network modules from some of the 75 pmax filtered tissue and stage subnetworks to show that i) these modules are enriched for genes that have context-relevant functions and ii) these modules are indeed context-relevant in that one does not recover the same module in different but related contexts. I validate the first point by first finding three modules and show that they all are enriched for context-relevant phenotypes and then point to examples of less characterized genes in these modules that have been shown to be context-relevant in recent studies in *Drosophila* and also in studies on orthologous counterparts in other model organisms and in humans. I demonstrate the second point by showing that an ovary-relevant module does not occur in similarly filtered early embryo networks even though the ovary and embryo relevant networks are enriched for similar phenotypes.

The example of the ovary module (Figure 3-15) has 32 genes that are expressed at greater than 75 pmax in the ovary and these genes have 169 interactions among them. The genes that have been annotated with the phenotype terms 'cell cycle defective' and 'female sterile' (colored blue) include the majority of genes in this module. There are also genes in this module about which little is known (colored yellow), while for some genes such as *Dsor1, Pi3K21B, CG31687* and *Nab2* (colored red) there is evidence that they may have ovary-relevant functions. For example, *Dsor1* is annotated with 'lethal' and 'neuroanatomy defective' in FlyBase but there is also evidence that it may play a role in ocyte maturation [174] and its ortholog in mammals has been found to cause

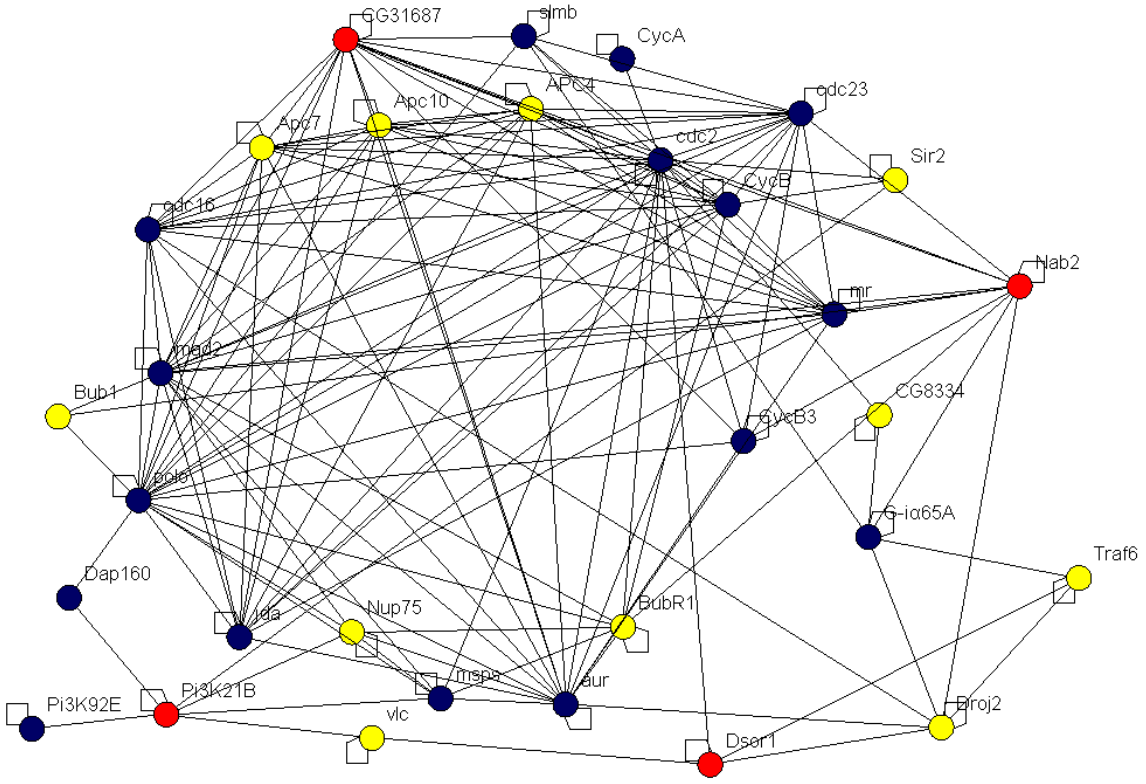


Figure 3-15 An ovary module. Network module made using IM Browser showing a set of genes that are expressed at greater than 75 pmax in the ovary. Genes with ovary-relevant phenotypes are colored in blue, genes with largely unknown phenotypes in yellow and putative ovary-relevant genes in red.

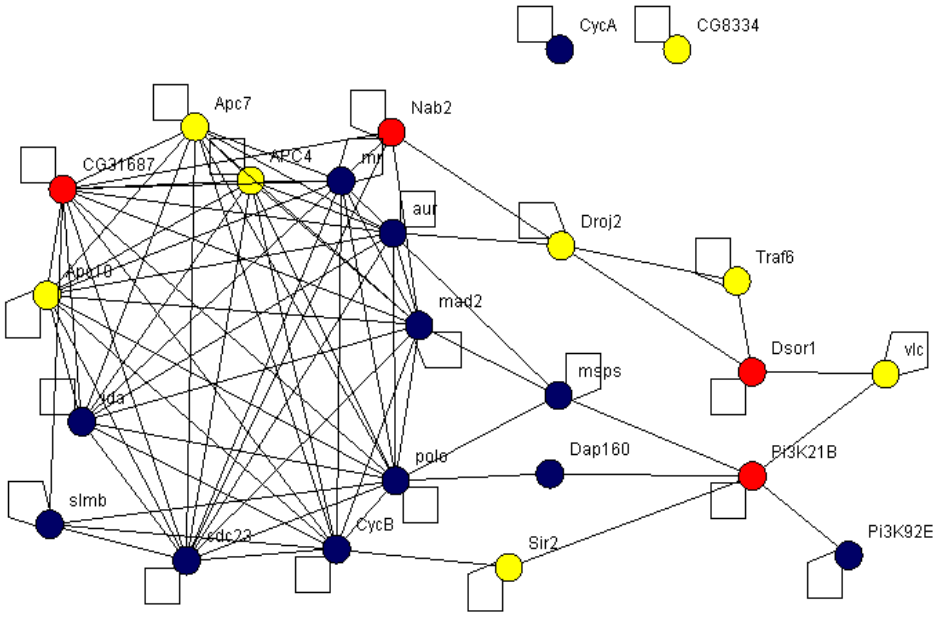


Figure 3-16 Ovary module in 0–2 hr embryos. Network module isolated from the greater than 75 pmax 0–2 hr embryo subnetwork using the ovary module genes as query genes. The color scheme is same as in ovary module.

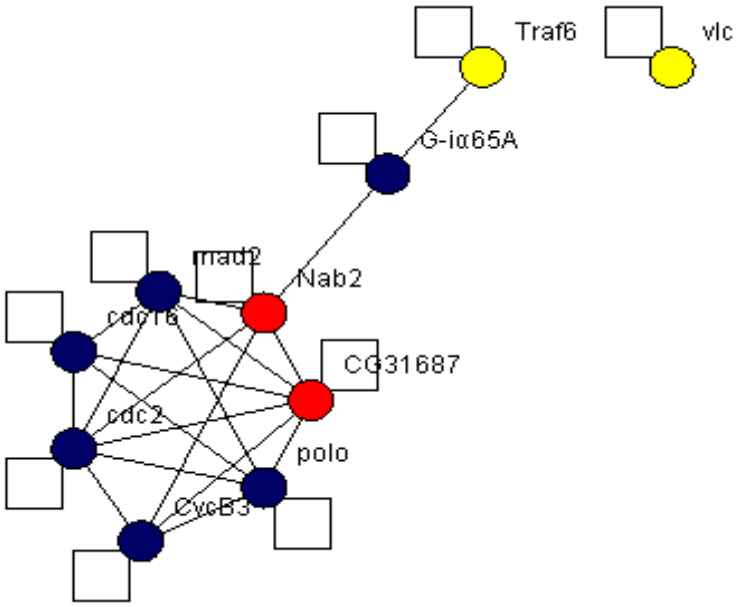


Figure 3-17 Ovary module in 2–4 hr embryo. Network module isolated from the greater than 75 pmax 2–4 hr embryo subnetwork using the ovary module genes as query genes. The color scheme is same as in ovary module.

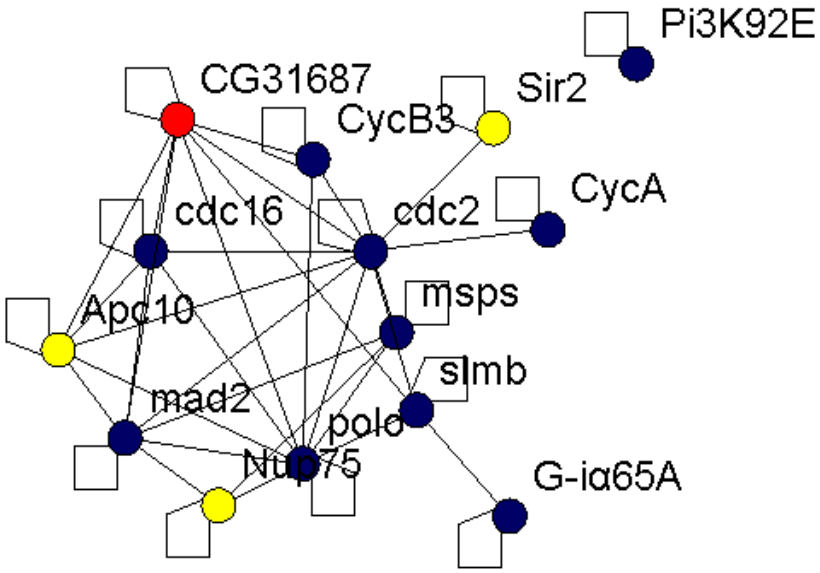


Figure 3-18 Ovary module in 4–6 hr embryo. Network module isolated from the greater than 75 pmax 4–6 hr embryo subnetwork using the ovary module genes as query genes. The color scheme is same as in ovary module.

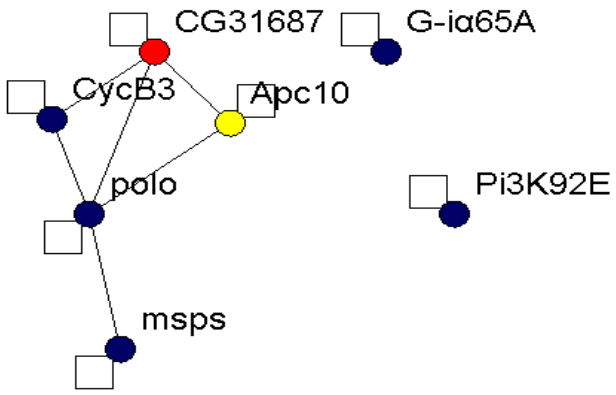


Figure 3-19 Ovary module in 6–8 hr embryo. Network module isolated from the greater than 75 pmax 6–8 hr embryo subnetwork using the ovary module genes as query genes. The color scheme is same as in ovary module.

developmental abnormalities and death in murine models due to cell proliferation defects and defects in apoptosis [175]; it has also been associated with a human disease (Cardiofaciocutaneous syndrome) [113]. The gene *Pi3K21B* is another example of a gene that has been shown to play a role in cell size and cell number regulation in imaginal discs and could potentially also play a role in the ovary [176]. The gene *CG31687* was identified in a recent screen for modifiers of planar cell polarity and also potentially relevant for oocyte maturation or early embryogenesis functions [177]. *Nab2* has been shown in a recent study to be an essential germline contribution to early embryogenesis [178]. I can therefore say that the 75 pmax filtered ovary module is predictive of ovary-relevant phenotypes.

The brain module includes 41 genes with 170 interactions among them (Figure 3-20). The genes that have been annotated with brain-relevant phenotype terms such as ' neuroanatomy defective', 'neurophysiology defective' and 'behavior defective' (colored in blue) include more than half of the genes. Again, the genes that are colored in yellowhave mostly unknown functions. Genes marked in red include *arm, Arf51F, CG9170, PICK1, CG18208, Ac3, Pkc98E* have evidence of brain-related functions. *Arf51F* was not yet annotated with a brain relavant phenotype, but has been shown recently to play a role in the behavioral response to ethanol [179]. *arm* is an essential gene causing embryonic lethality when disrupted, but mutations in the human orthologs, *CTNNB1*, have been shown to associate with severe intellectual disability in humans [180]. Another gene *CG9170* has a human ortholog *CEP164*, which has been shown to play a major role in ciliopathies which exhibit a broad range of phenotypes in many tissues including human

brain and retina. A recent study on the *Drosophila* gene *PICK1,* which came out well after I had initiated our studies is another example of the predictive power of our module. PICK1 is a transport protein and it has been shown that it is a modifier protein that can alleviate the symptoms of human neurodegenerative disorders such as Spinocerebellar ataxia [181]. The gene *CG18208* has a human ortholog namely *ADRA2A* which is an alpha adrenergic receptor. This gene has been associated with attention deficit hyperactivity disorder and some embryonic and placental abnormalities in humans and mice [113]. Another recent study on the gene *Ac3* was published after I had downloaded phenotype data from FlyBase. Alterations to this gene have been shown to result in defects in circadian and locomotor behaviors [182]. A seventh gene namely *Pkc98E* has been shown in an RNAi screen to play a role in neurite outgrowth [183]. In summary I show that the brain module represented here is predictive of genes with brain-relevant phenotypes.

The eye module includes 81 genes with 230 interactions among them (Figure 3-21). The genes annotated with eye-relevant phenotype terms 'visual behavior defective', 'eye color defective' and 'body color defective' are colored in blue and include fewer than half the genes in this module. The genes colored in yellow predominantly include those that do not yet have phenotype terms associated with them in Flybase. Two genes in this module namely *gish* and *didum* are not annotated with eye-relevant phenotypes. *gish* may have roles in learning and memory [184] and also play a role in eye disc formation [185]. Some genes in this module are annotated with 'lethal' in FlyBase, such as *Pp1-87B, flw, Hex-A, Fps85D, Cka. Fatp, sn,* 14-3-3-*ε, skpA* are annotated with 'lethal' and

'neuroanatomy defective' phenotypes; the first three of these were only recently found to be annotated with 'neuroanatomy defective' phenotype after I downloaded the phenotype terms. The genes marked in red (*CG5027, Fbxl4 and Pp1-87B)* show evidence that they have eye-relevant functions. The gene *CG5027* is orthologous to *TMX3* in human and *tmx3* in zebra fish. In humans, mutations in this gene have been associated with microphthalmia and reduced expression of this gene in zebra fish results in a small eye phenotype [186]. The gene *Fbxl4* has no phenotypic data available but has been shown to play a role in deactivation of rhodopsin-mediated signaling [187]. Another gene, *Pp1-87B,* is annotated with 'lethal' phenotype but not with an eye-relevant phenotype. Earlier studies have shown that reduced expression of this gene leads to learning and behavior abnormalities including visual conditioning [188].

Another interesting aspect to note in the three different tissue networks is the presence of two ubiquitously expressed genes, *Droj2* and *Dsor1*. Both genes are in the ovary module, while *Droj2* is in the brain module and *Dsor1* is in the eye module, each in different network contexts. Dsor1 plays roles in the Ras/ERK signaling pathway and it coordinates inputs from many pathways [189]. The gene *Droj2* is also represented in the eye module and it has been shown to have chaperone functions [190] and also play a role in small RNA-mediated gene silencing [191]. It would interesting to find the roles these genes play in the ovary and eye respectively.

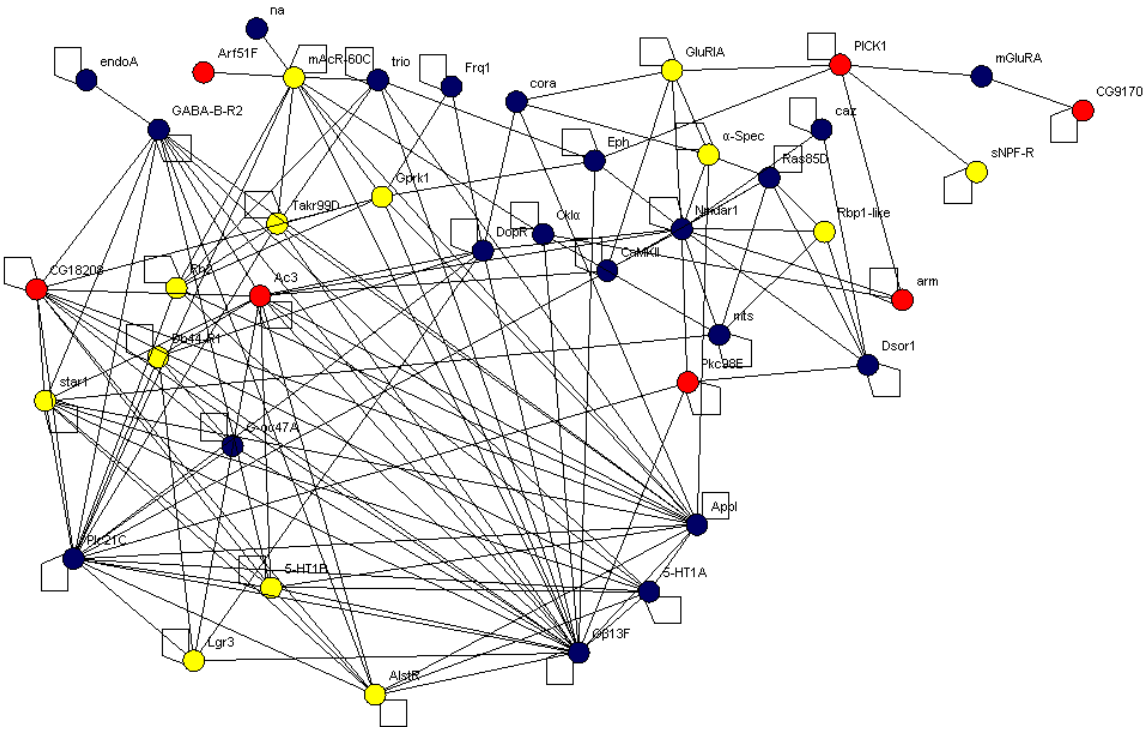


Figure 3-20 A brain module. Network module made using IM Browser showing a set of genes that are expressed at greater than 75 pmax in the brain. Genes with brain-relevant phenotypes are colored in blue, genes with largely unknown phenotypes in yellow and putative brain-relevant genes in red.

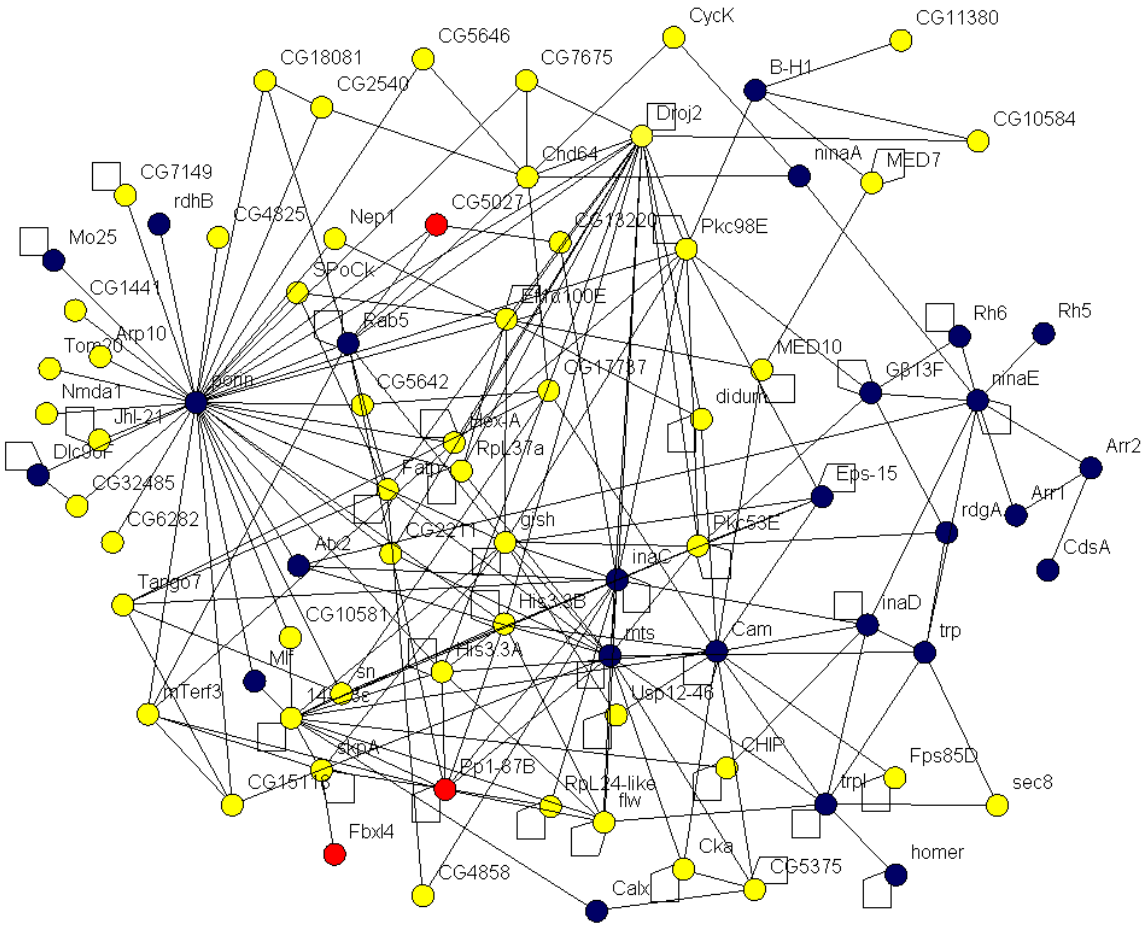


Figure 3-21 An eye module. Network module made using IM Browser showing a set of genes that are expressed at greater than 75 pmax in the eye. Genes with eye-relevant phenotypes are colored in blue, genes with largely unknown phenotypes in yellow and putative eye-relevant genes in red.

I took the same genes that form the ovary module (Figure 3-15) and searched in the embryo 0–2 hr, embryo 2–4 hr, embryo 4–6 hr and embryo 6–8 hr 75 pmax stage filtered networks to see if I could recover the same ovary module. The idea behind this exercise was to show that the same ovary network should not be returned even though both the ovary and the early embryo 75 pmax networks are enriched for similar phenotypes such as mitotic cell cycle defective and female sterile phenotypes. I was unable to recover the ovary module in the four early embryo networks and also the modules including the same query genes progressively became smaller (Figures 3-16 to 3-19). For example, the fractions of the ovary module that were recovered in the 75 pmax subnetworks were 60% for the in 0–2 hr embryo, 18% for the 2–4 hr embryo, 27% for the 4–6 hr embryo, and 8% for the 6–8 hr embryo. This pattern is consistent with maternal transcript depletion in the early embryo stages.

3.4 Discussion

In this study I used transcriptome data to examine the PPI and PDI interactomes of *Drosophila* and arrived at several general conclusions. First, I show that the ubiquitously expressed proteins interact among themselves significantly more than expected by random chance. Second, I have shown that the tissue and stage specific proteins interact with a core network of ubiquitously expressed proteins, potentially modifying it for tissue or developmental time point specific functions. Third, I show that the tissue and stage specific proteins rarely interact with each other directly or even indirectly through other non-specific proteins. These results for PPI are in agreement with previous human studies that have shown that the tissue-specific proteins have few

interactions among themselves, the ubiquitous proteins frequently interact with each other, and the tissue-specific proteins primarily interact with the ubiquitous proteins [150, 151]. In addition, I have shown that these results hold true for stage-specific and stage-ubiquitous proteins. In evolution it is seen that proteins are often repurposed rather than created anew. Therefore, the interactions of the tissue or stage specific proteins with the ubiquitous proteins may take place to recruit the ubiquitous network to perform context specific functions (some examples in [192-198]). This confirms the need and usefulness of the pmax scaling for filtering context relevant biological networks.

The next motivation of our study was to develop a method to filter context-relevant networks from the composite interactome using gene expression. Towards this aim I first developed a normalization procedure that scaled (on a percent scale) a gene's expression to its maximum expression in a given tissue or stage. In order to show that this method is more relevant than using an expression cut off in a tissue, I compared phenotype enrichment in gene sets filtered using the tissue percent maximum scale to those filtered based on the average expression of all genes in a tissue. I found that the pmax scaling performed significantly better, in that the filtered gene lists were significantly more enriched for tissue-relevant phenotypes than the average expression filtered gene lists. I then used this intuitive scale to identify interaction networks that might be relevant in different tissues or stages and found that the percent maximum filtered networks are indeed even more context-relevant by virtue of them being further enriched for the tissue or stage relevant phenotypes than even the corresponding filtered gene lists. With this result, I can state that irrespective of expression specificity, the

comparative level of expression of a gene is most likely responsible for phenotypes that are observed in a given tissue or stage. I then checked if filtering for context-relevant subnetworks or modules using the pmax scaling would allow one to predict the mutant phenotypes of unknown genes in the module. I show that such prediction is indeed possible using three examples of tissue-relevant modules with genes whose functions or phenotypes I was able to predict or verify relevancy.

Using the pmax scaling to scale gene expression allows one to scale a gene's expression on a scale of 1 to 100 irrespective of its physiological level of expression. I have shown that filtering genes or networks using this scale produces a context-relevant list of genes or subnetworks as evidenced by enrichment of tissue or stage relevant phenotypes in these networks. I have effectively combined qualitative interaction data and quantitative expression data in a systematic way thus providing a starting point to generate hypotheses about the functions of poorly characterized genes and to answer systems level questions. Judging from our interactome-transcriptome correlation results, there seems to be a tightly connected network in the ovary that gradually becomes sparser in the early embryo and then is pruned and adapted or customized in cells of various tissues as development progresses; the customization being the expression of various tissue-specific proteins. Consistent with this idea the percentmax filter at 75 or 50, identifies the most genes for the ovary, smaller numbers for early embryos but significantly fewer genes for later stages and tissues.

I show that the interactome networks when filtered based on the pmax gene expression filter generate biologically relevant subnetworks. As more detailed expression

data become available I expect such filtered networks to be even more relevant in future studies. For example, further studies to identify pathways from interaction networks is now possible with newly emerging expression data for many treatment conditions [83]. There are also new tissue and stage expression data with better space and time resolution becoming available. It will be interesting to analyze the interactome network with these new data in light of the findings from recent systems genetics approaches that have associated tissue and sex bias in transcript levels to complex traits in *Drosophila* [199, 200] and from proteome studies in humans that have shown that different cell types have cell-type specific differences in levels of ubiquitously expressed proteins [201]. Also, another interesting study would be to look at miRNA expression and the corresponding pmax expression of targets in the relevant tissues or disease conditions. This would further allow verifying our findings that expression level of a gene rather than specificity is a key determining factor for context-specific phenotypes.

CHAPTER 4 CONCLUSIONS AND FUTURE DIRECTIONS

4.1 Conclusions

In this thesis I have described the integration of heterogeneous interaction data for *Drosophila* into DroID, the *Drosophila* interactions database, making it a one-stop public resource for interaction data. I have also made it possible to filter the interaction data using gene expression data to generate context-relevant networks making DroID a one-of-a kind resource for biologists. In the two years since the upgraded DroID has been available, several studies have used the heterogeneous interaction data in DroID to advance our understanding of *Drosophila* biology thus validating the need for such a resource for biologists (examples [37, 71-76, 134, 135]). In addition to this, I have also identified organizing principles of interaction networks based on genome-wide gene expression data in the tissues and the entire life cycle of *Drosophila*. I have shown that all tissues and stages have a core PPI network of ubiquitously expressed proteins to which tissue and stage specific proteins attach to potentially modulate specific functions. In view of these organizing principles, I developed a normalized expression filter for interaction networks. I have shown that networks generated by using this filter are context-relevant evidenced by enrichment of the respective phenotypes. This filter has been implemented in DroID and I anticipate that studies on interactome networks using this filter will further our understanding of biology. Here I summarize my work on DroID and the interactome-transcriptome correlation.

4.1.1 DroID is a comprehensive resource for interactome analysis

In order to get a system-wide understanding of a biological process, interaction networks must be built by collecting and integrating heterogeneous data such as protein-protein, protein-DNA and RNA-gene interactions. Also, different genes are expressed in different cell types and at different time points and this in turn determines if an interaction can take place. Therefore interactome data must be integrated with gene expression data in order to build networks that are biologically relevant. Accordingly, a biologist must collect genomic data from gene-centered repositories, interaction data from various repositories including those for PPI, PDI and RRI, and integrate them. Next in order to make context-relevant networks, the interaction data must be filtered using expression data, which must again be collected and processed. This involves significant effort and time, thereby showing the need for a one-stop resource for biologists. Hence, DroID was first created to serve as a repository for all *Drosophila* PPIs and it provided interfaces to query and filter the interaction data. The understanding of the importance of integrating additional interaction types along with PPI led to the evolution of the database. I have developed DroID to serve not only as a comprehensive resource for *Drosophila* PPI data and have also made it a resource that allows combining PPI data with TF-gene, RNA-gene, and gene-gene data enabling the construction of more complete biological networks. I have also made it possible to integrate the multiple interaction types with the spatiotemporal gene expression data in DroID, thereby allowing users to explore interaction networks that are relevant to specific developmental times and tissues. DroID is the only resource that allows the combined analysis of heterogeneous interaction data and also the only resource that allows users to identify

context-relevant networks based on gene expression data. Currently, TF-gene and RNA-gene networks can be visualized as networks and also analyzed together with PPI only in DroID. As the amount and diversity of interaction data continue to increase, databases such as DroID that allow users to access and interpret comprehensive gene and protein interaction data will become essential tools for the study of biological pathways and networks. DroID is the only one of its kind for any organism and this void has to be filled for other organisms so that the vast amounts of emerging data can be used efficiently to gain insights into biological systems.

4.1.2 Examination of the interactome in different tissue and developmental contexts leads to novel insights on the organization of biological systems

In this study I examined the interactome of *Drosophila* in different tissue and developmental contexts and showed first that the ubiquitously expressed proteins interact among themselves significantly more than expected by random chance. Second, I have shown that the tissue and stage specific proteins interact with a core network of ubiquitously expressed proteins, potentially tailoring it for tissue or developmental time point specific functions. Third, I showed that the tissue and stage specific proteins rarely interact with each other directly or even indirectly through another non-specific protein. Fourth, I showed that non-ubiquitous TFs tend to regulate ubiquitous genes as often as they regulate non-ubiquitous targets. The results for the PPI studies have been shown for the first time in *Drosophila* and are in agreement with previous human studies [150, 151]. In addition, I have shown that these results hold true for stage-specific and stage-ubiquitous proteins. The interactions of the tissue or stage specific proteins with the

ubiquitous proteins may take place to recruit the ubiquitous network to perform context specific functions (some examples in [192-198]). This confirmed the need and usefulness of scaling gene expression data for identifying context relevant biological networks. Towards this aim I first developed a normalization procedure that scaled (on a percent scale) a gene's expression to its maximum expression in a given tissue or stage. In order to show that this method is more relevant than a cut-off based on the average expression of all genes in a tissue, I compared phenotype enrichment of filtered sets of tissue percent maximum expressed genes and tissue above average expressed genes. I found that the percent maximum scaling performed significantly better, in that the filtered gene lists were significantly more enriched for tissue-relevant phenotypes than the average expression filtered gene lists. I then used this intuitive scale to filter interaction networks that might be relevant in different tissues or stages and found that the percent maximum filtered networks are indeed even more context-relevant as demonstrated by their further enrichment for tissue or stage relevant phenotypes compared to the corresponding filtered gene lists. With this result, I can state that irrespective of expression specificity, the comparative level of expression of a gene is most likely responsible for phenotypes that are observed in a given tissue or stage. I then checked if filtering for context-relevant subnetworks or modules using the percent maximum scaling would allow one to predict the phenotypes of unknown genes. I show that such prediction is indeed possible using three examples of tissue-relevant modules with genes whose functions or phenotypes we were able to predict or verify relevancy.

4.2 Future directions

In this thesis I described early attempts at integrating transcriptome and interactome data to gain insights into the organization of biological networks. In this section I summarize what I see in the future for DroID and for interactome studies.

4.2.1 Impact of new interaction and expression data on DroID

DroID is an interaction database with an associated suite of tools that enable the analysis and filtering of interaction data. As new interaction or expression data become available DroID needs to be updated not only to keep current the data and tables, but also to ensure that the interfaces accommodate the new data. For example, DroID has several PPI tables that can be analyzed separately or in combination with other data sets. The PPI tables differ in the type of information that was collected about each interaction and also the methods used to detect interactions. As new PPI data sets become available a better option may be to have a master table with all the PPI and optionally provide another interface for a user needing to query subsets of the PPI separately or view all the experimental details. DroID also computes interologs for *Drosophila* by downloading interaction data for yeast worm and human from BioGRID IntAct and MINT [46-48] along with HPRD [202] and REACTOME [203] for human; a high quality resource that could be included to the list is the BIND database [114]. A new feature of DroID is the TF-gene data, which include data from the REDfly database [52] and from modENCODE project [51]. The modENCODE project (www.modencode.org) is ongoing and is poised to generate more TF-gene data [18, 204], which have to be periodically added to DroID. Another new feature is the RNA-gene table which includes data from TargetScanFly

[126] and MinoTar [61] databases and the modENCODE project [51]. All the RNA-gene data in DroID are from prediction algorithms, so expanding the sources to include information on experimental verification of miR and their targets as it becomes available, for example, from existing curation efforts (see, for example, TarBase; [57]) would add value to the RNA-gene interactions. The DroID database and its associated interfaces are capable of accommodating additional functional information about interactions such as whether or not an interaction represents an activating or repressing relationship. This information will become increasingly available and its incorporation into DroID will make it more useful for biologists. Another feature unique to DroID is the availability of filters for interactions based on raw or normalized expression data. As new expression data such as gene expression values under different treatment conditions become available (www.modencode.org), new tables have to be constantly added and the interfaces have to accommodate that. These data would add new dynamic components to the interaction data sets in DroID.

Another area of expansion and development for DroID is to make the interaction data compatible with formats that are community standards. DroID needs to expand support for the Protein Standards Initiative Molecular Interaction (PSI-MI) format [205]. DroID tables include fields compatible with PSI-MI, including the PSI-MI ontology terms and identifiers whenever they are available for an interaction. DroID also maintains the proposed minimal information required to report a molecular interaction [206], when that information is available for interactions. Currently users can download DroID data and convert them to the PSI-MI format using one of the publically available tools for that

purpose (e.g., see www.psidev.info). DroID currently has three different interfaces that allow filtering of genes and interaction networks and also a download area that allows the download of the filtered interactions. Interactions in different data sets can be downloaded in their entirety, and also the entire DroID database including confidence scores and gene expression correlations in bulk. However, DroID cannot be accessed programmatically. Programmatic access is routinely provided by large PPI databases such as IntAct, MINT and BioGRID [46-48]. Such access to DroID data will be useful for many applications as users requiring querying and filtering of large volumes of data quickly can do it in a single step, such as for example systems biology tools and algorithms. An interface for such downloads was developed by HUPO Proteomics Standards Initiative termed PSICQUIC and stands for Proteomics Standards Initiative Common Query Interface [207] and has become a community standard for programmatic access to interaction repositories. New data also necessitate constant updates to maintenance scripts used to make regular updates to DroID. Currently DroID obtains gene attribute data from FlyBase which itself changes its data structure to accommodate new expression data. Automated confidence scoring [106] of interactions during every DroID update would also add value to the interactions in DroID. The correlation data that I have calculated for all gene pairs along with normalized expression data tables would also be useful downloads to the user community.

Having stated the possible routes for future development of DroID, I would also like to point out that there is no other resource such as DroID that is available for any organism including human. Although, a PPI database for human that allows the querying

of interaction data along with a tissue-wide gene expression filter has recently become available [208], its query interfaces and the network visualization tool are not as user-friendly as DroID. The resource also has made available a limited gene expression filter based on human tissue expression data but the filter is similarly not intuitive. The resource also does not allow the integration of other heterogeneous interaction data. Therefore there is an immediate need to fill the one-stop organism-specific public resource void for other model organisms and human to enable efficient use of the torrents of data that are becoming available. DroID would be a useful database to emulate for such endeavors. This would be especially useful for human biologists and biomedical researchers as there are not only vast amounts data but also more types of data for human, for example disease causing variants and polymorphisms.

4.2.2 Impact of new and emerging expression data on interactome studies

Correlation of the transcriptome and interactome has generated insights into the organization of interaction networks and has also shown that the level of expression of a gene is important for tissue or stage identity. There are several other areas for exploration correlating the interactome and transcriptome. For example, in my studies of the PDI network, I found that non-ubiquitous TFs may play a role in regulating the level of ubiquitous gene expression (data not shown). For example, the *CycY* gene is up-regulated in tissues such as larval CNS and ovary where it is regulated by TFs that have restricted patterns of expression. *CycY* is down-regulated significantly in testis where it is regulated by a testis-specifc TF. In other tissues where *CycY* is neither up or down-regulated, it is

regulated only by ubiquitous TFs. This is significant in that the levels of expression of ubiquitous genes are potentially regulated in different tissues and stages which may result in a rewiring of the networks for specific functions [209]. This could potentially rewire PPI networks for context-relevant functions. Exploring this area would provide clues as to how tissues are maintained and how development is orchestrated by non-ubiquitous TFs to rewire the interaction networks for specific functions. For this analysis the normalized gene expression will very be useful as changes in levels of gene expression in different tissues or stages have to be considered.

Another way that the integrated interactome and transcriptome can be used is to more easily identify members of pathways. For example, new expression data for various treatment conditions are available for *Drosophila* from the modENCODE project. The expression changes in response to different treatment conditions can be utilized in the following manner. The maximally expressed gene networks before and after the treatment can be compared to each other. This method might be much more tractable than looking at the entire tissue-expressed network. This method can be extended to studies of gene expression in diseased tissues also. First, the percent maximum filter must be applied to make maximally expressed networks in the normal or untreated condition. Next the treated tissue expression data should be combined to make a new percent of maximum filter and applied to the interaction data. By looking for which gene(s) have been removed from or added to the before-treatment network along with its connections would provide clues to the members of pathways that change dynamically or the network that was perturbed in disease.

On a final note, the level of transcribed RNA has been used as a proxy for protein abundance in gene expression analyses. Various studies have measured the correlation of transcript versus protein quantity in light of the fact that protein expression is regulated not only at the level of transcription but also at the level of translation (RNA stability and translation efficiency) and post translation (protein stability and turnover, other modifications) [210]. A study comparing gene expression by RNA-seq and quantitative proteomics found weak but positive correlation between transcript and protein abundance overall and this is not totally unexpected as there are many levels and mechanisms of post-transcriptional regulation known [167]. Other recent studies also found moderate correlation between RNA-seq measurements and protein measurements using mass spectrometry. The same studies also found moderate correlation between microarray and RNA-seq measurements [211, 212]. Quantitative proteomic data are expected to be increasingly available in the near future and this would perhaps help make the interactome data even more context-relevant. In the future, more proteomic data will be available and it could be used the same way that I have used the transcriptome data such as in the percent maximum scale.

The thoughts of initiating such studies in the future are very exciting and make me wonder what we are going to learn about biological systems in the not too distant future.

APPENDIX A

The scripts below can be run in any order or in parallel except the last script which has to be run at the end and produces output for the next set of scripts.

get_flybase_gene_attr.pl

```
#!/usr/bin/perl

#

# a simple script for connecting to the flybase

# PostgreSQL database.

#

use DBI;

use strict;


my $dbh;

my $sth;

my @vetor;

my %g_int = ();


open (STDERR, ">c_error.out");

open (OUT2, ">concat.txt");

open (OUT1, ">sec_fbgn.txt");

open (OUT3, ">secCG_Prifbgn.txt");

open (GEN, ">gen_int.txt");

open (CGS, ">Fbgn_cg.txt");


my %HoA = ();

#my $pat = qr/.*?FBgn\d+.*?/os;

#oracle date format for current date DD-MMM-YY

my $curr_day = (localtime)[3];

my @mon = ("JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", "DEC");

my $curr_mon = (localtime)[4];
```

```
my $curr_year = (localtime)[5]+1900;

my $dte= "$curr_day-";

$dte .=$mon[$curr_mon];

$dte .= "-";

$dte .= substr $curr_year, 2;

print "$dte\n";


$dbh = DBI->connect("DBI:Pg:dbname=flybase; host=flybase.org;port=5432","flybase", "flybase");


if ($dbh) {

  print "connected\n";


  $sth = $dbh->prepare("SELECT DISTINCT uniquename, synonym_sgml FROM feature f, feature_synonym fs,
synonym s, cvterm cvt, cvterm cvt2  WHERE f.type_id = cvt.cvterm_id AND cvt.name = 'gene' AND f.feature_id =
fs.feature_id AND fs.synonym_id = s.synonym_id AND fs.is_current = 't' AND fs.is_internal = 'f' AND s.type_id =
cvt2.cvterm_id AND cvt2.name = 'symbol' AND f.is_obsolete = 'f' AND f.uniquename LIKE 'FBgn%' AND
f.organism_id =1;");

  $sth->execute;

          #fbgn and synonym_sgml for dmel only

  while (@vetor = $sth->fetchrow) {

    my $fbgn = $vetor[0];

    my $sym = $vetor[1];


  if (exists $HoA{$fbgn}) {push @{ $HoA{$fbgn} }, $sym;}

  else {$HoA {$fbgn} = [ $sym ]; }

  }

  print "records:", $sth->rows(), "\n";

  $sth->finish;

  my $size = scalar(keys %HoA);

  print "got symbols: $size\n";


 #fbgn and fullname
```

```
  $sth = $dbh->prepare("SELECT DISTINCT uniquename, synonym_sgml from feature f, cvterm cvt,
feature_synonym fs, synonym s, cvterm cvt2 WHERE f.type_id = cvt.cvterm_id AND f.organism_id = 1 and cvt.name
= 'gene' AND f.uniquename like 'FBgn%' AND f.feature_id = fs.feature_id AND fs.synonym_id = s.synonym_id AND
fs.is_internal = 'f' AND s.type_id = cvt2.cvterm_id AND cvt2.name = 'fullname' AND fs.is_current = 't' AND
f.is_obsolete = 'f';");

  $sth->execute;

  @vetor = ();

  while (@vetor = $sth->fetchrow) {

    my $fbgn = $vetor[0];

    my $fullname = $vetor[1];


  if (exists $HoA{$fbgn}) {  $HoA{$fbgn}[1] = $fullname;}

  }

   print "records:", $sth->rows(), "\n";

  $sth->finish;


  $size = scalar(keys %HoA);

  print "got fullnames: $size\n";


  my $furl = "http://flybase.net/reports/";

  my $j;

  foreach $j (keys %HoA){

    my $some = $HoA{$j}[1];

    $HoA{$j}[2] = "$furl$j.html";

    if (defined($some)){;}

    else {

          $HoA{$j}[1] = "";

#         print "found null\t";

    }

  }
```

```
#fbgn and secondary fbgn

#- will need to output just fbgn-secfbgn for updating interaction tables


$sth = $dbh->prepare("SELECT f.uniquename, accession as SecondaryFBgn  FROM feature f, feature_dbxref fd,
dbxref d, db  WHERE f.feature_id = fd.feature_id AND fd.dbxref_id = d.dbxref_id AND fd.is_current = 'f' AND
f.uniquename like 'FBgn%' AND d.db_id = db.db_id AND db.name = 'FlyBase' AND organism_id =1 AND
f.is_obsolete = 'f';");

$sth->execute;


@vetor = ();

my %hoa = ();

my $CGpat = qr/^(C[GR]\d+)/os;

while (@vetor = $sth->fetchrow) {

   my $fbgn = $vetor[0];

   my $sec = $vetor[1];

   print OUT1 "$fbgn\t$sec\n";

   if (exists $hoa{$fbgn}) {push @{ $hoa{$fbgn} } , $sec;}

   else {$hoa {$fbgn} = [ $sec ]; }

}

 print "records:", $sth->rows(), "\n";

$sth->finish;

$size = scalar(keys %HoA);

print "got secondary fbgn: $size\n";


#one fbgn can have many secondary fbgns, so need two hashes

my $i;

foreach $i (keys %hoa){

        my $fbgn = $i;

    my $secf = join(", ", @{ $hoa{$i} });

    if (exists $HoA{$fbgn}) {  $HoA{$fbgn}[3] = $secf;}

 }

$j = "";
```

```
  foreach $j (keys %HoA){
    my $some = $HoA{$j}[3];
    if (defined($some)){;}
    else {$HoA{$j}[3] = "";}
  }

# gene class
  $sth = $dbh->prepare("SELECT f.uniquename, fp.value as gene_class FROM feature f, featureprop fp, cvterm cvt
WHERE f.feature_id = fp.feature_id AND fp.type_id = cvt.cvterm_id AND cvt.name = 'promoted_gene_type' AND
f.uniquename LIKE 'FBgn%' AND f.organism_id = 1;");
  $sth->execute;

  @vetor = ();
  while (@vetor = $sth->fetchrow) {
    my $fbgn = $vetor[0];
    my $genecls = $vetor[1];

  if (exists $HoA{$fbgn}) {  $HoA{$fbgn}[4] = $genecls;}
  }
   print "records:", $sth->rows(), "\n";
  $sth->finish;
  $size = scalar(keys %HoA);
  print "got gene class: $size\n";
  $j = "";
  foreach $j (keys %HoA){
    my $some = $HoA{$j}[4];
    if (defined($some)){;}
    else {$HoA{$j}[4] = "";}
  }

# CG symbol
```

```
  $sth = $dbh->prepare("SELECT uniquename, accession  FROM feature f, feature_dbxref fd, dbxref d, db WHERE f.feature_id = fd.feature_id AND fd.dbxref_id = d.dbxref_id AND fd.is_current = 't' AND d.db_id = db.db_id AND db.name = 'FlyBase Annotation IDs' AND f.uniquename LIKE 'FBgn%' AND f.organism_id = 1 AND f.is_obsolete = 'f'");


   $sth->execute;


   @vetor = ();

   while (@vetor = $sth->fetchrow) {

      my $fbgn = $vetor[0];

      my $cg = $vetor[1];

      print CGS "$fbgn\t$cg\n";

      if (exists $HoA{$fbgn}) {  $HoA{$fbgn}[5] = $cg;}

   }

    print "records:", $sth->rows(), "\n";

   $sth->finish;

   $size = scalar(keys %HoA);

   print "got cg symbol: $size\n";

   $j = "";

   foreach $j (keys %HoA){

      my $some = $HoA{$j}[5];

      if (defined($some)){;}

      else {$HoA{$j}[5] = "";}

   }


#name and symbol synonyms


$sth = $dbh->prepare("SELECT DISTINCT f.uniquename, s.name AS synonym  FROM cvterm cvt, feature f, feature_synonym fs, synonym s, pub p  WHERE f.feature_id = fs.feature_id AND fs.synonym_id = s.synonym_id AND fs.pub_id = p.pub_id AND s.type_id = cvt.cvterm_id AND (cvt.name = 'symbol' OR cvt.name = 'fullname') AND fs.is_current = 'f' AND f.uniquename LIKE 'FBgn%' AND f.organism_id =1 AND f.is_obsolete = 'f';");


$sth->execute;
```

```
@vetor = ();
%hoa = ();
while (@vetor = $sth->fetchrow) {
   my $fbgn = $vetor[0];
   my $syn = $vetor[1];
   if (exists $hoa{$fbgn}) {push @{ $hoa{$fbgn} } , $syn;}
   else {$hoa {$fbgn} = [ $syn ]; }
   if ($syn=~ $CGpat){
     print OUT3 "$fbgn\t$1\n";
   }
}
 print "records:", $sth->rows(), "\n";
$sth->finish;
$size = scalar(keys %HoA);
print "got synonyms: $size\n";

$i="";
foreach $i (keys %hoa){
     my $fbgn = $i;
     my $synf = join(", ", @{ $hoa{$i} });
     if (exists $HoA{$fbgn}) {  $HoA{$fbgn}[6] = $synf;}
 }
$j = "";
foreach $j (keys %HoA){
   my $some = $HoA{$j}[6];
   if (defined($some)){;}
   else {$HoA{$j}[6] = "";}
}
```

```
#interpro domains


    $sth = $dbh->prepare("SELECT DISTINCT f.uniquename, accession, d.description FROM feature f, feature_dbxref
fd, dbxref d, db WHERE f.feature_id = fd.feature_id AND fd.dbxref_id = d.dbxref_id AND fd.is_current = 't' AND
d.db_id = db.db_id AND db.name = 'INTERPRO' AND f.uniquename LIKE 'FBgn%' AND f.organism_id = 1;");


  $sth->execute;


  @vetor = ();

  %hoa = ();

  while (@vetor = $sth->fetchrow) {

     my $fbgn = $vetor[0];

     my $acc = $vetor[1];

     if (length ($acc) <1 ){ next; }

     my $desc = $vetor[2];

     if (length ($desc) <1 ){ next; }

     if (exists $hoa{$fbgn}) {push @{ $hoa{$fbgn} } , "$acc=$desc";}

     else {$hoa {$fbgn} = [ "$acc=$desc" ]; }

  }

   print "records:", $sth->rows(), "\n";

  $sth->finish;

  $size = scalar(keys %HoA);

  print "got interpro domains: $size\n";


  $i="";

  foreach $i (keys %hoa){

       my $fbgn = $i;

       my $ipr = join(", ", @{ $hoa{$i} });

       if (exists $HoA{$fbgn}) {  $HoA{$fbgn}[7] = $ipr;}

   }

 $j = "";
```

```
  foreach $j (keys %HoA){
    my $some = $HoA{$j}[7];
    if (defined($some)){;}
    else {$HoA{$j}[7] = "";}
  }

# go MF

  $sth = $dbh->prepare("SELECT DISTINCT f.uniquename, c.name, d.accession FROM   feature f, feature_cvterm fc, cvterm c, db, dbxref d, cv WHERE  f.feature_id = fc.feature_id and fc.cvterm_id = c.cvterm_id  and  c.cv_id = cv.cv_id and cv.name = 'molecular_function'  and  c.dbxref_id = d.dbxref_id and d.db_id = db.db_id and db.name = 'GO'  and  f.uniquename LIKE 'FBgn%';");
  $sth->execute;

  @vetor = ();
  %hoa = ();
  while (@vetor = $sth->fetchrow) {
    my $fbgn = $vetor[0];
    my $acc = $vetor[2];
    if (length ($acc) <1 ){ next; }
    my $desc = $vetor[1];
    if (length ($desc) <1 ){ next; }
    if ($desc eq 'molecular_function'){ next;}
    if (exists $hoa{$fbgn}) {push @{ $hoa{$fbgn} } , "$acc=$desc";}
    else {$hoa {$fbgn} = [ "$acc=$desc" ]; }
  }
   print "records:", $sth->rows(), "\n";
  $sth->finish;
  $size = scalar(keys %HoA);
  print "got go molecular function: $size\n";

  $i="";
```

```
  foreach $i (keys %hoa){
      my $fbgn = $i;
      my $mf = join(", ", @{ $hoa{$i} });
      if (exists $HoA{$fbgn}) {  $HoA{$fbgn}[8] = $mf;}
   }
 $j = "";
  foreach $j (keys %HoA){
    my $some = $HoA{$j}[8];
    if (defined($some)){;}
    else {$HoA{$j}[8] = "";}
  }


# go BP


  $sth = $dbh->prepare("SELECT DISTINCT f.uniquename, c.name, d.accession FROM   feature f, feature_cvterm fc,
cvterm c, db, dbxref d, cv WHERE  f.feature_id = fc.feature_id and fc.cvterm_id = c.cvterm_id  and  c.cv_id =
cv.cv_id and cv.name = 'biological_process'  and  c.dbxref_id = d.dbxref_id and d.db_id = db.db_id and db.name =
'GO'  and  f.uniquename LIKE 'FBgn%';");

  $sth->execute;


  @vetor = ();
  %hoa = ();
  while (@vetor = $sth->fetchrow) {
    my $fbgn = $vetor[0];
    my $acc = $vetor[2];
    if (length ($acc) <1 ) { next; }
    my $desc = $vetor[1];
    if (length ($desc) <1 ){ next; }
    if ($desc eq 'biological_process'){ next;}
    if (exists $hoa{$fbgn}) {push @{ $hoa{$fbgn} } , "$acc=$desc";}
    else {$hoa {$fbgn} = [ "$acc=$desc" ]; }
  }
```

```
 print "records:", $sth->rows(), "\n";
 $sth->finish;
 $size = scalar(keys %HoA);
 print "got go biological process: $size\n";

 $i="";
 foreach $i (keys %hoa){
     my $fbgn = $i;
     my $bp = join(", ", @{ $hoa{$i} });
     if (exists $HoA{$fbgn}) {  $HoA{$fbgn}[9] = $bp;}
  }
$j = "";
 foreach $j (keys %HoA){
   my $some = $HoA{$j}[9];
   if (defined($some)){;}
   else {$HoA{$j}[9] = "";}
 }

# go CC

  $sth = $dbh->prepare("SELECT DISTINCT f.uniquename, c.name, d.accession FROM   feature f, feature_cvterm fc,
cvterm c, db, dbxref d, cv WHERE  f.feature_id = fc.feature_id and fc.cvterm_id = c.cvterm_id  and  c.cv_id =
cv.cv_id and cv.name = 'cellular_component'  and  c.dbxref_id = d.dbxref_id and d.db_id = db.db_id and db.name =
'GO'  and  f.uniquename LIKE 'FBgn%';");
 $sth->execute;

 @vetor = ();
 %hoa = ();
 while (@vetor = $sth->fetchrow) {
   my $fbgn = $vetor[0];
   my $acc = $vetor[2];
   if (length ($acc) <1 ){ next; }
```

```
    my $desc = $vetor[1];
    if (length ($desc) <1 ){ next; }
    if ($desc eq 'cellular_component'){ next;}
    if (exists $hoa{$fbgn}) {push @{ $hoa{$fbgn} } , "$acc=$desc";}
    else {$hoa {$fbgn} = [ "$acc=$desc" ]; }
  }
   print "records:", $sth->rows(), "\n";

  $sth->finish;
  $size = scalar(keys %HoA);
  print "got go cellular component: $size\n";

  $i="";
  foreach $i (keys %hoa){
      my $fbgn = $i;
      my $cc = join(", ", @{ $hoa{$i} });
      if (exists $HoA{$fbgn}) {  $HoA{$fbgn}[10] = $cc;}
   }
 $j = "";
  foreach $j (keys %HoA){
    my $some = $HoA{$j}[10];
    if (defined($some)){;}
    else {$HoA{$j}[10] = "";}
  }

  my $i1;

  foreach $i1 (keys %HoA){
#       print join(", ", @{ $HoA{$i} }), "\n";
       print OUT2 "$i1\t", join("\t", @{ $HoA{$i1} }), "\t$dte\n";
```

```
  }


  $size = scalar(keys %HoA);

  print "size of hash: $size\n";


# get genetic interactions also - don't get 'unattributed' i.e. get ones that have a ref

# the order by clause is very important

  $sth = $dbh->prepare("SELECT g.uniquename, a.uniquename, p.uniquename  FROM feature g, cvterm cvt, feature a,
feature_relationship_pub frpb, pub p, feature_relationship fr LEFT OUTER JOIN feature_relationshipprop frp ON
(fr.feature_relationship_id = frp.feature_relationship_id)  WHERE g.feature_id = subject_id AND object_id =
a.feature_id AND fr.type_id = cvt.cvterm_id AND cvt.name = 'interacts_genetically' AND  fr.feature_relationship_id =
frpb.feature_relationship_id AND frpb.pub_id = p.pub_id AND  g.uniquename LIKE 'FBgn%' AND g.organism_id = 1
AND p.uniquename LIKE 'FBrf%' AND g.is_obsolete = 'f' AND a.is_obsolete = 'f' ORDER BY g.uniquename;");

  $sth->execute;


  @vetor = ();

  %hoa = ();

  my $url = "http://flybase.net/reports/";

  # remove dups even while getting the data

  while (@vetor = $sth->fetchrow) {

    my $fbgn = $vetor[0];

    my $gn;

    if (exists $HoA {$fbgn}){

       $gn = $HoA{$fbgn}[0]; #get symbol

    } else {next;}

    my $fbgn1 = $vetor[1];

    my $an;

    if (exists $HoA {$fbgn}){

       $an = $HoA{$fbgn1}[0]; #get symbol from the first hash

    } else {next;}

    my $pnm = $vetor[2];

    my $forw = "$fbgn\t$gn\t$fbgn1\t$an\t$pnm";

    my $rev = "$fbgn1\t$an\t$fbgn\t$gn\t$pnm";
```

```
    if ((exists $hoa {$forw})||(exists $hoa {$rev})) {next;}
    else { $hoa {$forw} = [ "$url$pnm.html" ]};
  }
   print "records:", $sth->rows(), "\n";

  $sth->finish;

  print "got genetic interactions\n";

  $i="";
  foreach $i (keys %hoa){
      my @fk = split (/\t/, $i);
      my $ref = pop( @fk );
#       print "$ref\t";
#           my @a = split (/\t/, $fk);
      my $f1 = $fk[0];
      my $f2 = $fk[2];
      my $s1 = $fk[1];
      my $s2 = $fk[3];
#      my $p = $a[4];
           my $nkey;
      if ($f2 lt $f1){
                    $nkey = "$f2\t$s2\t$f1\t$s1";
           }
           else{
                $nkey = join("\t", @fk);
           }
      my $purl = join(", ", @{ $hoa{$i} });
      if (exists $g_int{$nkey}) {
            my $nref = $g_int {$nkey}[0];
```

```
        $nref .= ", $ref";
        $g_int {$nkey}[0] = $nref;
        my $npurl = $g_int {$nkey}[1];
        $npurl .= ", $purl";
        $g_int {$nkey}[1] = $npurl;
      }
      else {
        $g_int {$nkey}[0] =  $ref;
        $g_int {$nkey}[1] = $purl;
       }
   }


 $dbh->disconnect();
} else {
  print "Cannot connect to Postgres server: $DBI::errstr\n";
  print " db connection failed\n";
}

my $j;
foreach $j ( keys %g_int) {
#           my @a = split (/\t/, $j);
#           my $f1 = $a[0];
#           my $f2 = $a[2];
#           my $s1 = $a[1];
#           my $s2 = $a[3];
#           my $p = $a[4];
#           if ($f2 lt $f1){
#                       print GEN "$f2\t$s2\t$f1\t$s1\t", join ("\t", @{ $g_int{$j} }), "\t$dte\tgg\n";
#           }
```

```
#		else {
#			print GEN "$j\t", join ("\t", @{ $g_int{$j} }), "\t$dte\tgg\n";
#		}
}

close STDERR;
close OUT2;
close OUT1;
close GEN;
close OUT3;
```

get_orthologs_plus.pl

```
#!usr/local/bin/perl
#
#:vim syntax=perl
#
use strict;

open (STDERR, ">inp_error.out");
open (OUT2, ">inp_hsp.txt");
open (OUT3, ">orth_tblp.txt");
open (SAC,">inp_scp.txt");
open (CEL, ">inp_cep.txt");
use Thila::Remove_dups;

my $b;
my $xline;
my $pos;
my %horth;
```

```
#for interologs make 3 separate files with hash keys as gn\tscr; push fbgn,fbpp,scr for each key and input to psicquic script

#for orthologs make gene key for hash with symbols; make another fbgn as hash key; push gene1,sym1,gene2,sym2 etc if sym is available

#for anonymous array creation depending on order make as ["","","",$hs] else push @{{}}[3] = $hs etc...



#patterns to search for and respective files

my $pat=qr/<CLUSTER CLUSTERNO\=\"\d+\" BITSCORE\=\"\d+\">.*?<\/CLUSTER>/os;

my $getpat=qr/<CLUSTER CLUSTERNO\=\"(\d+)\" BITSCORE\=\"(\d+)\">/os;

my $fbgn = qr/<GENE PROTID\=\"(FBpp\d+)\" GENEID\=\"(FBgn\d+)\" SCORE\=\"(.*?)\"
SPECIES\=\"D\.melanogaster\".*?\/>/os;

my $scf= "InParanoid.D.melanogaster-S.cerevisiae.xml";

my $scpat = qr/<GENE PROTID\=\"([Y|Q].*?)\" GENEID\=\"(.*?)\" SCORE\=\"(.*?)\"
SPECIES\=\"S\.cerevisiae\".*?\/>/os;

my $cef = "InParanoid.C.elegans-D.melanogaster.xml";

my $cepat = qr/<GENE PROTID\=\"(CE\d+)\" GENEID\=\"(WBGene\d+)\" SCORE\=\"(.*?)\"
SPECIES\=\"C\.elegans\".*?\/>/os;

my $drf = "InParanoid.D.melanogaster-D.rerio.xml";

my $drpat = qr/<GENE PROTID\=\"(ENSDARP\d+)\" GENEID\=\"(ENSDARG\d+)\" SCORE\=\"(.*?)\"
SPECIES\=\"D\.rerio\".*?\/>/os;

my $xtf = "InParanoid.D.melanogaster-X.tropicalis.xml";

my $xtpat = qr/<GENE PROTID\=\"(ENSXETP\d+)\" GENEID\=\"(ENSXETG\d+)\" SCORE\=\"(.*?)\"
SPECIES\=\"X\.tropicalis\".*?\/>/os;

my $mmf = "InParanoid.D.melanogaster-M.musculus.xml";

my $mmpat = qr/<GENE PROTID\=\"(ENSMUSP\d+)\" GENEID\=\"(ENSMUSG\d+)\" SCORE\=\"(.*?)\"
SPECIES\=\"M\.musculus\".*?\/>/os;

my $hsf = "InParanoid.D.melanogaster-H.sapiens.xml";

my $hspat = qr/<GENE PROTID\=\"(ENSP\d+)\" GENEID\=\"(ENSG\d+)\" SCORE\=\"(.*?)\"
SPECIES\=\"H\.sapiens\".*?\/>/os;

# symbol files

my $drsym = "Danio_rerio.gene_info"; #ncbi

my $hssym = "Homo_sapiens.gene_info"; #ncbi

my $cesym = "mart_export_wb_uniprot.txt";#wormbase biomart

my $mmsym = "Mus_musculus.gene_info"; #ncbi
```

```
my $scsym = "SGD_features.txt"; #SGD
my $xtsym = "xenopus_tropicalis_gene_info_ensembl.txt"; #ensembl biomart


#print "$pat\n";
#read ~16k chunks at a time from the XML file
        open (XMLF, "<$scf") or die "Can't open file $scf: $!";
        #open file in binary mode
        binmode(XMLF);
        open (SYMS, "<$scsym") or die "Can't open file $scsym: $!";
        my %hsc;
        while (<SYMS>){
                my $z= $_;
                chomp $z;
                my @a =  split(/\t/, $z);
                my $y = $a[3];
                my $s = $a[4];
                $hsc{$y} = $s;
        }
        while(read(XMLF,$b,16394)) {
                $xline .= $b;
#               print "here $xline\n\n\n";
        #check if xline has the starting <cluster> and the ending </cluster> tags
                while ($xline =~ /($pat)/g ){
                        my $inp=$1;
                        my $clustid;
                        my $bitscr;
                        my $fb;
                        my $fp;
                        my $scr;
#               print "$inp\n\n\n\n\n";
```

```
if ($inp =~ /$getpat/g){
        $clustid = $1;
        $bitscr = $2;
#       print "clust: $clustid, bit: $bitscr\n";
}
my $inp1 = $inp;
my $concaten;
while ($inp1=~ /$fbgn/g ) {
        $fp = $1;
        $fb = $2;
        $scr = $3;
#       print "fb: $fb\n";
        my $posf = pos($inp1);
        $inp1=substr($inp1,$posf);
        while ($inp=~ /$scpat/g){
                my $enp = $1;
                my $enp1;
                if (exists $hsc{$enp}){
                        $enp1 = "$enp(";
                        $enp1 .= $hsc{$enp};
                        $enp1 .= ")";
                }else {$enp1 = $enp; }
                my $en = $2;
                my $scr1 = $3;
                print SAC "$clustid\t$bitscr\t$fb\t$fp\t$scr\t$enp\t$en\t$scr1\n";
                if (length ($concaten) >0){ $concaten .= ","; }
                $concaten .= $enp1;
        }
        my @dups_arr = split (/,/, $concaten);
        my @no_dups = Remove_dups::rem_dups(@dups_arr);
```

```
                                        $concaten = join(",", @no_dups);
                                $horth{$fb} = [ $concaten ]; #***one fbgn belongs to only one clust
                                }
                                $pos = pos($xline);
                        }
                        $xline=substr($xline,$pos);

                }
close XMLF;
close SYMS;
        open (XMLF, "<$cef") or die "Can't open file $cef: $!";
        #open file in binary mode
        binmode(XMLF);
        open (SYMS, "<$cesym") or die "Can't open file $cesym: $!";
    my %hce;
         while (<SYMS>){
        my $z= $_;
        chomp $z;
        my @a =  split(/\t/, $z);
        my $y = $a[0];
        my $s = $a[1];
                if ($y =~ /(WBGene\d+)/){
                        my $in = $1;
                $hce{$in} = $s;
#                       print "$in\t$s\n";
                }
    }

        while(read(XMLF,$b,16394)) {
                $xline .= $b;
```

```
#                   print "here $xline\n\n\n";
                    while ($xline =~ /($pat)/g ){
                            my $inp=$1;
                            my $clustid;
                            my $bitscr;
                            my $fb;
                            my $fp;
                            my $scr;
#                   print "$inp\n\n\n\n\n";
                            if ($inp =~ /$getpat/g){
                                    $clustid = $1;
                                    $bitscr = $2;
#                           print "clust: $clustid, bit: $bitscr\n";
                            }
                            my $inp1 = $inp;
                            my $concaten;
                            while ($inp1=~ /$fbgn/g ) {
                                    $fp = $1;
                                    $fb = $2;
                                    $scr = $3;
#                           print "fb: $fb\n";
                                    my $posf = pos($inp1);
                                    $inp1=substr($inp1,$posf);
                                    while ($inp=~ /$cepat/g){
                                            my $enp = $2;
                                            my $enp1;
                        if (exists $hce{$enp}){
                            $enp1 = "$enp(";
                            $enp1 .= $hce{$enp};
                            $enp1 .= ")";
```

```
                    }else {$enp1 = $enp; }
                    my $en = $1;
                    my $scr1 = $3;
                    print CEL "$clustid\t$bitscr\t$fb\t$fp\t$scr\t$enp\t$en\t$scr1\n";
                    if (length ($concaten) >0){ $concaten .= ","; }
                    $concaten .= $enp1;
                                }
                                my @dups_arr = split (/,/, $concaten);
                my @no_dups = Remove_dups::rem_dups(@dups_arr);
                $concaten = join(",", @no_dups);
                if (exists $horth{$fb}) {  $horth{$fb}[1] = $concaten ;}
                                else { $horth {$fb}= [ "", $concaten ]; }#***one fbgn belongs to only one clust
                        }
                        $pos = pos($xline);
                }
                $xline=substr($xline,$pos);
        }
close XMLF;
close SYMS;
        open (XMLF, "<$drf") or die "Can't open file $drf: $!";
        #open file in binary mode
        binmode(XMLF);
        open (SYMS, "<$drsym") or die "Can't open file $drsym: $!";
    my %hdr;
         while (<SYMS>){
        my $z= $_;
        chomp $z;
        my @a =  split(/\t/, $z);
        my $y = $a[5];
        my $s = $a[2];
```

```
                if ($y =~ /(ENSDARG\d+)/){
            my $in = $1;
                $hdr{$in} = $s;
#                       print "$in\t$s\n";
                }
    }

        while(read(XMLF,$b,16394)) {
                $xline .= $b;
        #check if xline has the starting <cluster> and the ending </cluster> tags
                while ($xline =~ /($pat)/g ){
                        my $inp=$1;
                        my $clustid;
                        my $bitscr;
                        my $fb;
                        my $fp;
                        my $scr;
#               print "$inp\n\n\n\n\n";
                        if ($inp =~ /$getpat/g){
                                $clustid = $1;
                                $bitscr = $2;
                        }
                        my $inp1 = $inp;
                        my $concaten;
                        while ($inp1=~ /$fbgn/g ) {
                                $fp = $1;
                                $fb = $2;
                                $scr = $3;
#                       print "fb: $fb\n";
                                my $posf = pos($inp1);
```

```
                                        $inp1=substr($inp1,$posf);
                                        while ($inp=~ /$drpat/g){
                                                my $enp = $2;
                                                my $en = $1;
                                                my $scr1 = $3;
                                                my $enp1;
                        if (exists $hdr{$enp}){
                                $enp1 = "$enp(";
                                $enp1 .= $hdr{$enp};
                                $enp1 .= ")";
                        }else {$enp1 = $enp; }
                        if (length ($concaten) >0){ $concaten .= ","; }
                        $concaten .= $enp1;
#                                       print "$clustid\t$bitscr\t$fb\t$en\n";
                                        }
                                        my @dups_arr = split (/,/, $concaten);
                my @no_dups = Remove_dups::rem_dups(@dups_arr);
                $concaten = join(",", @no_dups);
                if (exists $horth{$fb}) {  $horth{$fb}[2] = $concaten ;}
                else { $horth {$fb}= [ "","",$concaten ]; }#***one fbgn belongs to only one clust


                        }
                        $pos = pos($xline);
                }
                $xline=substr($xline,$pos);
        }
close XMLF;
close SYMS;
        open (XMLF, "<$xtf") or die "Can't open file $xtf: $!";
        #open file in binary mode
```

```
	binmode(XMLF);
	open (SYMS, "<$xtsym") or die "Can't open file $xtsym: $!";
my %hxt;
	 while (<SYMS>){
	my $z= $_;
	chomp $z;
	my @a =  split(/\t/, $z);
	my $y = $a[0];
	my $s = $a[1];
		if ($y =~ /(ENSXETG\d+)/){
	my $in = $1;
	$hxt{$in} = $s;
		#	print "$in\t$s\n";
		}
}

	while(read(XMLF,$b,16394)) {
		$xline .= $b;
#		print "here $xline\n\n\n";
	#check if xline has the starting <cluster> and the ending </cluster> tags
		while ($xline =~ /($pat)/g ){
			my $inp=$1;
			my $clustid;
			my $bitscr;
			my $fb;
			my $fp;
			my $scr;
#		print "$inp\n\n\n\n\n";
			if ($inp =~ /$getpat/g){
				$clustid = $1;
```

```
                $bitscr = $2;
#               print "clust: $clustid, bit: $bitscr\n";
                }
                my $inp1 = $inp;
                my $concaten;
                while ($inp1=~ /$fbgn/g ) {
                        $fp = $1;
                        $fb = $2;
                        $scr = $3;
#               print "fb: $fb\n";
                        my $posf = pos($inp1);
                        $inp1=substr($inp1,$posf);
                        while ($inp=~ /$xtpat/g){
                                my $enp = $2;
                                my $en = $1;
                                my $scr1 = $3;
                                my $enp1;
        if (exists $hxt{$enp}){
            $enp1 = "$enp(";
            $enp1 .= $hxt{$enp};
            $enp1 .= ")";
        }else {$enp1 = $enp; }
        if (length ($concaten) >0){ $concaten .= ","; }
        $concaten .= $enp1;
                        }
                        my @dups_arr = split (/,/, $concaten);
    my @no_dups = Remove_dups::rem_dups(@dups_arr);
    $concaten = join(",", @no_dups);
    if (exists $horth{$fb}) {  $horth{$fb}[3] = $concaten ;}
    else { $horth {$fb}= [ "", "", "", $concaten ]; }#***one fbgn belongs to only one clust
```

```
                        }
                        $pos = pos($xline);
                }
                $xline=substr($xline,$pos);
        }
close XMLF;
close SYMS;
        open (XMLF, "<$mmf") or die "Can't open file $mmf: $!";
        #open file in binary mode
        binmode(XMLF);
        open (SYMS, "<$mmsym") or die "Can't open file $mmsym: $!";
    my %hmm;
         while (<SYMS>){
        my $z= $_;
        chomp $z;
        my @a =  split(/\t/, $z);
        my $y = $a[4];
                my $y1 = $a[5];
        my $s = $a[2];
        if (($y =~ /(ENSMUSG\d+)/) || ($y1 =~ /(ENSMUSG\d+)/)){
            my $in = $1;
            $hmm{$in} = $s;
#             print "$in\t$s\n";
        }
    }

        while(read(XMLF,$b,16394)) {
                $xline .= $b;
#               print "here $xline\n\n\n";
        #check if xline has the starting <cluster> and the ending </cluster> tags
```

```
while ($xline =~ /($pat)/g ){
        my $inp=$1;
        my $clustid;
        my $bitscr;
        my $fb;
        my $fp;
        my $scr;
#       print "$inp\n\n\n\n\n";
        if ($inp =~ /$getpat/g){
                $clustid = $1;
                $bitscr = $2;
#       print "clust: $clustid, bit: $bitscr\n";
        }
        my $inp1 = $inp;
        my $concaten;
        while ($inp1=~ /$fbgn/g ) {
                $fp = $1;
                $fb = $2;
                $scr = $3;
                my $posf = pos($inp1);
                $inp1=substr($inp1,$posf);
                while ($inp=~ /$mmpat/g){
                        my $enp = $2;
                        my $enp1;
      if (exists $hmm{$enp}){
        $enp1 = "$enp(";
        $enp1 .= $hmm{$enp};
        $enp1 .= ")";
      }else {$enp1 = $enp; }
      my $en = $1;
```

```
                        my $scr1 = $3;
                        if (length ($concaten) >0){ $concaten .= ","; }
                        $concaten .= $enp1;
#                                       print "$clustid\t$bitscr\t$fb\t$en\n";
                                        }
                                        my @dups_arr = split (/,/, $concaten);
                    my @no_dups = Remove_dups::rem_dups(@dups_arr);
                    $concaten = join(",", @no_dups);
                    if (exists $horth{$fb}) {  $horth{$fb}[4] = $concaten ;}
                    else { $horth {$fb}= [ "", "", "", "", $concaten ]; }#***one fbgn belongs to only one clust
                            }
                            $pos = pos($xline);
                    }
                    $xline=substr($xline,$pos);

                }
close XMLF;
close SYMS;
            open (XMLF, "<$hsf") or die "Can't open file $hsf: $!";
            #open file in binary mode
            binmode(XMLF);
            open (SYMS, "<$hssym") or die "Can't open file $hssym: $!";
        my %hhs;
            my %hhs1; #mim num
            my %hhs2; # gene desc
             while (<SYMS>){
            my $z= $_;
            chomp $z;
            my @a =  split(/\t/, $z);
            my $y = $a[5];
```

```
        my $s = $a[2];
                my $dsc= $a[8];
                my $in;
                if ($y =~ /(ENSG\d+)/){
                        $in = $1;
                        $hhs{$in} = $s;
#                       print "$in\t$s\n";
                }
                if ($y =~ /(MIM:\d+)/){
                        my $m = $1;
#                       print "$m\n";
                        $hhs1{$in} = $m;
                        $hhs2{$in} = $dsc;
                }
    }

        while(read(XMLF,$b,16394)) {
                $xline .= $b;
        #check if xline has the starting <cluster> and the ending </cluster> tags
                while ($xline =~ /($pat)/g ){
                        my $inp=$1;
                        my $clustid;
                        my $bitscr;
                        my $fb;
                        my $fp;
                        my $scr;
#                       print "$inp\n\n\n\n\n";
                        if ($inp =~ /$getpat/g){
                                $clustid = $1;
                                $bitscr = $2;
```

```
#                       print "clust: $clustid, bit: $bitscr\n";
                        }
                        my $inp1 = $inp;
                        my $concaten; my $concaten1; my $concaten2;
                        while ($inp1=~ /$fbgn/g ) {
                                $fp = $1;
                                $fb = $2;
                                $scr = $3;
#                       print "fb: $fb\n";
                                my $posf = pos($inp1);
                                $inp1=substr($inp1,$posf);
                                while ($inp=~ /$hspat/g){
                                        my $enp = $2;
                                        my $enp1; my $enp2; my $enp3;
                if (exists $hhs{$enp}){
                        $enp1 = "$enp(";
                        $enp1 .= $hhs{$enp};
                        $enp1 .= ")";
                }else {$enp1 = $enp; }
                                                #08.22.11
                                        if (exists $hhs1{$enp}) {
                                                $enp2 = $hhs1{$enp};
                                        }else{$enp2 = "";}
                                        if (exists $hhs2{$enp}){
                                                $enp3 = $hhs2{$enp};
                                        }else{$enp3 = "";}
                my $en = $1;
                my $scr1 = $3;
                print OUT2 "$clustid\t$bitscr\t$fb\t$fp\t$scr\t$enp\t$en\t$scr1\n";
                if (length ($concaten) >0){ $concaten .= ","; }
```

```
                    $concaten .= $enp1;
                    if (length ($concaten1) >0){ $concaten1 .= ","; }
                    $concaten1 .= $enp2;
                    if (length ($concaten2) >0){ $concaten2 .= ","; }
                    $concaten2 .= $enp3;
#                                   print "$clustid\t$bitscr\t$fb\t$en\n";
                                    }
                                    my @dups_arr = split (/,/, $concaten);
                my @no_dups = Remove_dups::rem_dups(@dups_arr);
                $concaten = join(",", @no_dups);
                                    my @dups_arr1 = split (/,/, $concaten1);
                my @no_dups1 = Remove_dups::rem_dups(@dups_arr1);
                $concaten1 = join(",", @no_dups1);
                                    my @dups_arr2 = split (/,/, $concaten2);
                my @no_dups2 = Remove_dups::rem_dups(@dups_arr2);
                $concaten2 = join(",", @no_dups2);
                if (exists $horth{$fb}) {
                                        $horth{$fb}[5] = $concaten ;
                                        $horth{$fb}[6] = $concaten1;
                                        $horth{$fb}[7] = $concaten2;
                                    }
                else { $horth {$fb}= [ "", "", "", "", "", $concaten, $concaten1, $concaten2 ]; }#***one fbgn
belongs to only one clust
                            }
                            $pos = pos($xline);
                }
                $xline=substr($xline,$pos);
        }
close XMLF;
close SYMS;
```

```
my $j;

foreach $j ( keys %horth) {

        print OUT3 "$j\t", join ("\t", @{ $horth{$j} }), "\n";

}

close OUT2;

close OUT3;

close STDERR;

close CEL;

close SAC;
```

map_sec_pri_fbgn.pl

```
#!/usr/bin/perl
#
# input file is sec_fbgn.txt from flybase download script
# having 1st col as primary fbgn and the 2nd col as seccondary fbgn
use strict;

open (STDERR, ">c_error.out");
open (OUT2, ">sec_pri_fbgn_noupd.txt");
open (OUT1, ">sec_pri_fbgn_upd.txt");

open(INP, "<$ARGV[0]")  or die("Cannot open file '$ARGV[0]' for reading\n");
my %hcgs = ();

while (<INP>){
        my $fbgn = $_;
        chomp $fbgn;
#       print "$fbgn\n";
        my @a = split (/\t/, $fbgn);
        my $fb= $a[0];
        my $cg = $a[1];
                #this check is weird but necessary due to flybase quirks
        if ((exists $hcgs{$cg}) && ( $hcgs{$cg} ne $fb)){
                        $hcgs{$cg} .= ", $fb";
                }
                else { $hcgs {$cg} =  $fb;}
}
my $ky;
foreach $ky (keys %hcgs){
                my $o = $hcgs{$ky};
                #print OUT2 "$ky\t$o\n";
                if (length ($o) > 11){
                        print OUT2 "$ky\t$o\n";
                                delete $hcgs{$ky};
                }
}

my $ky1;
```

```
foreach $ky1 (keys %hcgs){
        print OUT1 "$ky1\t$hcgs{$ky1}\n";
}

close INP;
close OUT1;
close OUT2;
close STDERR;
```

APPENDIX B

The scripts below can be run in any order or in parallel.

get_other_phy_mapfb_0113.pl

```
#!usr/local/bin/perl

#

#:vim syntax=perl

#

#this script uses the psicquic tool to get data from Biogrid, intact and MINT using MIQL syntax

# read in fly FBgn, CG numbers from flybase downloads and store as hash to match CGs from these dbs to FBgns and output the files with FBgns

# for intact no mapping is required - just grab fbgn - this is the final version of the other physical script.

open (STDERR, ">error.out");

open (OTHER, ">other_phy13.txt");

open (FINAL, ">other_physical_interactions13.txt");

open(INP, "<$ARGV[0]")  or die("Cannot open file '$ARGV[0]' for reading\n"); #fbgn_cg.txt

open(INS, "<$ARGV[1]")  or die("Cannot open file '$ARGV[1]' for reading\n"); #secCG_prifgbn.txt

use strict;

use LWP::Simple;

use Thila::Remove_dups;

#use Thila::Sort_array;

my $upd = "sec_pri_fbgn_upd.txt";

my $noupd = "sec_pri_fbgn_noupd.txt";

open (NOUP, "<$noupd");

open (UPD, "<$upd");

my $bind = "testing_curated_bind.txt";

open (BIND, "<$bind");

open (UNP, "<fbgn_uniprottrembl.txt");

open (UNPP, "<fbgn_uniprot.txt");


open (BIOGRID, "<BIOGRID-ORGANISM-Drosophila_melanogaster-3.2.96.mitab.txt");
```

```
my @b;
my %hpmid_ignore = (        15710747=>'i',
                            15575970=>'i',
                            18840285=>'i',
                            14605208=>'i',
                            19079254=>'i',
                            14613974=>'i',
                            16603075=>'i',
                            22036573=>'i',
                            22028469=>'i');
my %hcgs1 = ();
my %hcgs2 = ();
my %hint_db = ();
my %hint_pid = ();
my %hint_met = ();

my $url = "http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=retrieve&db=pubmed&list_uids=";

while (<INP>){
        my $fbgn = $_;
        chomp $fbgn;
#       print "$fbgn\n";
        my @a = split (/\t/, $fbgn);
        my $fb= $a[0];
        my $cg = $a[1];
        # be sure the input file has primary fb to primary CG one to one mapping
    $hcgs1 {$cg} =  $fb;
}

while (<INS>){
```

```
	my $fbgn = $_;
	chomp $fbgn;
#	 print "$fbgn\n";
	my @a = split (/\t/, $fbgn);
	my $fb= $a[0];
	my $cg = $a[1];
	#this check is weird but necessary due to flybase quirks
	if ((exists $hcgs2{$cg}) && ( $hcgs2{$cg} ne $fb)){
		$hcgs2{$cg} .= ", $fb";
	}
	else { $hcgs2 {$cg} =  $fb;}
}
my $k1;
foreach $k1 (keys %hcgs2){
	my $o = $hcgs2{$k1};
	if (length ($o) > 11){
		delete $hcgs2{$k1};
	}
}

my %hprot;
while (<UNP>){
	my $fbgn = $_;
	chomp $fbgn;
#	 print "$fbgn\n";
	my @a = split (/\t/, $fbgn);
	my $fb= $a[0];
	my $up = $a[1];
	if ((exists $hprot{$up}) && ( $hprot{$up} ne $fb)){
		$hprot{$up} .= ", $fb";
```

```
        }
        else { $hprot{$up} =  $fb;}
}
#my $k2;
#foreach $k2 (keys %hprot){
#        my $o = $hprot{$k2};
#        if (length ($o) > 11){
#                delete $hprot{$k2};
#        }
#}
while (<UNPP>){
        my $fbgn = $_;
        chomp $fbgn;
#       print "$fbgn\n";
        my @a = split (/\t/, $fbgn);
        my $fb= $a[0];
        my $up = $a[1];
        if ((exists $hprot{$up} ) && ( $hprot{$up} ne $fb)){
                $hprot{$up} .= ", $fb";
        }
        else { $hprot{$up} =  $fb;}
}
my $k3;
foreach $k3 (keys %hprot){
        my $o = $hprot{$k3};
                if ($k3 =~ /Q7KN55/) {print "$hprot{$k3}\n"; }
        if (length ($o) > 11){
                delete $hprot{$k3};
        }
}
```

```
close UNP;
close UNPP;

my %hupd=();
my %hnoup = ();
my @d;
while (<NOUP>){
	chomp $_;
		@d = split(/\t/, $_);
		my $fb = $d[0];
		my $pris = $d[1];
	$hnoup{$fb} = $pris;
}
close NOUP;

while (<UPD>){
	chomp $_;
		@d = split(/\t/, $_);
		my $fb = $d[0];
		my $pri = $d[1];
	$hupd{$fb} = $pri;
}
close UPD;

my $curr_day = (localtime)[3];
my @mon = ("JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", "DEC");
my $curr_mon = (localtime)[4];
my $curr_year = (localtime)[5]+1900;
my $dte= "$curr_day-";
$dte .=$mon[$curr_mon];
```

```
$dte .= "-";
$dte .= substr $curr_year, 2;
print "$dte\n";

my $bl;
my $il;
my $ml;
my $content = ();
my @allint_arr;

my @rec;
while (<BIND>){
        chomp $_;
        @rec = split(/\t/, $_);
        my $pubid = $rec[4];
        my $fb1 = $rec[0];
        my $fb2 = $rec[1];
        my $method=$rec[5];
        if (exists $hnoup{$fb1}){next;}
        if (exists $hnoup{$fb2}){next;}
        if (exists $hupd{$fb1}){ $fb1 = $hupd{$fb1}; }
        if (exists $hupd{$fb2}){ $fb2 = $hupd{$fb2}; }
        my $str_int = "$fb1\t$fb2\t$rec[2]\t$rec[3]\t$method\t$pubid\tBIND-$dte\t$dte";
        print OTHER "$str_int\n";
        push @allint_arr , $str_int ;

}

# biogrid get all physical interactions (MI OBO term 'association')
# AND both belonging to Dmel (7227)
```

```
# AND minus droid data in pubids listed

while (<BIOGRID>){

	chomp $_;

	my $c = $_;

	$c =~ s/entrez gene\///g;

	$c =~ s/"//g;

	$c =~ s/psi-mi://g;

	push @b , $c;

}

#type to keep col 11 direct interaction and physical association

my $direc = "direct interaction";

my $assoc = "physical association";

#$content = get("http://tyerslab.bio.ed.ac.uk:8080/psicquic-
ws/webservices/current/search/query/type:%22*association%22%20AND%20taxidA:7227%20AND%20taxidB:7227
%20AND%20NOT%20pubid:%20(15710747%20OR%2015575970%20OR%2018840285%20OR%2014605208%20
OR%2019079254%20OR%2014613974%20OR%2016603075%20OR%2022036573%20OR%2022028469)");

#die "Couldn't get it!" unless defined $content;


print "getting biogrid\n";

#my $patb = qr/Dmel(CG\d+)/os; # had to change AGAIN as BioGRID format changed - 08.14.2011

my $patb = qr/locuslink:Dmel.*?(CG\d+)/os;

my $patpub = qr/pubmed:(\d+)/os;

my $patpub_unp = qr/pubmed:(\w+.*?)/os;


#$content =~ s/entrez gene\///g;

#$content =~ s/"//g;

#$content =~ s/psi-mi://g;


#my @b = split (/\n/, $content);

#fields 3, 4 sym, 5, 6 CG, 7 meth, 9 pmid, add date download, db

foreach $bl (@b){

	my @in = split (/\t/, $bl);
```

```
my $id1; my $id2;
my $pubid;
my $fb1; my $fb2;
my $method;
my $type = $in[11];

if (($type =~ /$direc/) || ($type =~ /$assoc/)){ ; }
else { next; }
#	print $in[2], "\n";
if ($in[2] =~ $patb) {
    $id1 = $1;
    #     print "$id1\n";
}
else {next;}
if ($in[3] =~ $patb){
    $id2= $1;
}
else {next;}
if ($in[8] =~ $patpub || $in[8] =~ $patpub_unp){
        $pubid = $1;
        #       print "$pubid\n";
} else {next;}
if (exists $hpmid_ignore{$pubid}){ next; }

$method = $in[6];
#if ($in[4] =~ $patb) {
#     $id1 = $1;
#}
#	else {next;}
#	if ($in[5] =~ $patb){
```

```
#               $id2= $1;
#           }
#           else {next;}

# select FBgn from Primary first and then Secondary CG list
            if ((exists $hcgs1{$id1}) && (exists $hcgs1{$id2})){
                        $fb1 = $hcgs1{$id1};
                        $fb2 = $hcgs1{$id2};
            }
            elsif((exists $hcgs1{$id1}) && (exists $hcgs2{$id2})){
                        $fb1 = $hcgs1{$id1};
            $fb2 = $hcgs2{$id2};
            }
            elsif((exists $hcgs2{$id1}) && (exists $hcgs1{$id2})){
                        $fb1 = $hcgs2{$id1};
            $fb2 = $hcgs1{$id2};
            }
            elsif((exists $hcgs2{$id1}) && (exists $hcgs2{$id2})){
                        $fb1 = $hcgs2{$id1};
            $fb2 = $hcgs2{$id2};
      } else {next;}
            $in[2] =~ s/locuslink://g;
            $in[3] =~ s/locuslink://g;
            my $str_int = "$fb1\t$fb2\t$in[2]\t$in[3]\t$in[6]\t$pubid\tBioGRID-$dte\t$dte";
            print OTHER "$str_int\n";
            push @allint_arr , $str_int ;
}

#intact data
#format changed again - so modified code again - 12.24.12
```

```
#$content = ();

#$content =
get("http://www.ebi.ac.uk/Tools/webservices/psicquic/intact/webservices/current/search/query/type:%22*association%
22%20AND%20taxidA:7227%20AND%20taxidB:7227%20AND%20NOT%20pubid:%20(15710747%20OR%20155
75970%20OR%2018840285%20OR%2014605208%20OR%2019079254%20OR%2014613974%20OR%2016603075
%20OR%2022036573%20OR%2022028469)");

#die "Couldn't get it!" unless defined $content;

#print "$content";

#$content =~ s/uniprotkb://g;

#$content =~ s/"//g;

#$content =~ s/psi-mi://g;


print "getting intact\n";

my $pati0 = qr/Dmel_(CG\d+)\(orf name/os;

my $pati = qr/(CG\d+)\(orf name/os;

#my $pati0 = qr/flybase:(FBgn\d+)/os;

open (INTACT, "<intact012413.txt");

#my @i = split (/\n/, $content);

#foreach $il (@i) {

#       print "$il\n";

while (<INTACT>){

        chomp $_;

        $il = $_;

        $il =~ s/uniprotkb://g;

        $il =~ s/"//g;

        $il =~ s/psi-mi://g;


#       print "$il\n";

        my @in = split (/\t/, $il);

    my $id1; my $id2;

        my $altid1 = $in[0]; my $altid2 = $in[1];

#       if ($altid1 =~ /Q7KN55/ || $altid2 =~ /Q7KN55/) {
```

```
#                       print "$il\n";
#                       print "$hprot{$altid1}\t$hprot{$altid2}\n";
#           }
#           print "$hprot{$altid1}\t$hprot{$altid2}\n";
        my $pubid;
        my $fb1; my $fb2;
        my $method;
#           print "$in[4]\n";
        if ($in[4] =~ $pati0) {
            $id1 = $1;
        }
            elsif ($in[4] =~$pati){
                $id1 = $1;
            }
            elsif (exists $hprot{$altid1}){;}
            else {next;}

        if ($in[5] =~ $pati0){
            $id2= $1;
        }
            elsif ($in[5] =~$pati){
            $id2 = $1;
            }
            elsif (exists $hprot{$altid2}) {;}
            else {next;}
#           print "$id1\t$id2\n";
        if ($in[8] =~ $patpub || $in[8] =~ $patpub_unp){
            $pubid = $1;
                        if (exists $hpmid_ignore{$pubid}){ next; }
```

```
                #               if ($pubid =~ 15710747|| $pubid =~ 15575970 || $pubid =~ 18840285 || $pubid =~
14605208 || $pubid =~ 19079254 || $pubid =~ 14613974 || $pubid =~ 16603075 || $pubid =~ 22036573 || $pubid =~
22028469) { next;}

        } else {next;}


                $method = $in[6];

#               if ($in[19] =~ $pati0){

#                       $fb1 = $1;

#               }else {next;}

#               if ($in[20] =~$pati0){

#                       $fb2 = $1;

#               }else {next;}

#               if (exists $hnoup{$fb1}){next;}

#               if (exists $hnoup{$fb2}){next;}

#               if (exists $hupd{$fb1}){ $fb1 = $hupd{$fb1}; }

#               if (exists $hupd{$fb2}){ $fb2 = $hupd{$fb2}; }


        if ((exists $hcgs1{$id1}) && (exists $hcgs1{$id2})){

                $fb1 = $hcgs1{$id1};

                $fb2 = $hcgs1{$id2};

        }

        elsif((exists $hcgs1{$id1}) && (exists $hcgs2{$id2})){

                $fb1 = $hcgs1{$id1};

                $fb2 = $hcgs2{$id2};

        }

        elsif((exists $hcgs2{$id1}) && (exists $hcgs1{$id2})){

                $fb1 = $hcgs2{$id1};

                $fb2 = $hcgs1{$id2};

        }

        elsif((exists $hcgs2{$id1}) && (exists $hcgs2{$id2})){

                $fb1 = $hcgs2{$id1};
```

```
            $fb2 = $hcgs2{$id2};
        }
            elsif ((exists $hprot{$altid1} )&& (exists $hprot{$altid2} )){
                    $fb1 = $hprot{$altid1};
                    $fb2 = $hprot{$altid2};
                    print "$fb1\t$fb2\n";
            }else {next;}
#print "$fb1\t$fb2\n";
            my $str_int = "$fb1\t$fb2\t$in[0]\t$in[1]\t$in[6]\t$pubid\tIntAct-$dte\t$dte";
        print OTHER "$str_int\n";
            push @allint_arr , $str_int ;
}


# fields 3, 4 (CG), 5, 6 (sym), 7 (meth), 9 pmid, date, db


# MINT - gets all interaction data - type doesn't work - so need to make sure to only get 'association' type
$content = ();
$content =
get("http://mint.bio.uniroma2.it/mint/psicquic/webservices/current/search/query/taxidA:7227%20AND%20taxidB:722
7%20AND%20NOT%20pubid:%20(15710747%20OR%2015575970%20OR%2018840285%20OR%2014605208%20
OR%2019079254%20OR%2014613974%20OR%2016603075%20OR%2022036573%20OR%2022028469)");
die "Couldn't get it!" unless defined $content;


print "getting mint\n";
my $patm = qr/(CG\d+)\(orf name/os;


$content =~ s/uniprotkb://g;
$content =~ s/"//g;
$content =~ s/psi-mi://g;


my @m = split (/\n/, $content);
my $rem_pat = qr/MI:0403\(colocalization\)/os; #field 12
```

```
# MINT doesn't have genetic interactions - so need to only remove colocalization
# fields 5, 6 (CG), 7 (meth), 9 pmid, date, db
foreach $ml (@m) {
        if ($ml =~ $rem_pat) {next;}
        my @in = split (/\t/, $ml);
    my $id1; my $id2;
    my $pubid;
    my $fb1; my $fb2;
    my $method;

    if ($in[4] =~ $patm) {
        $id1 = $1;
    } else {next;}

    if ($in[5] =~ $patm){
        $id2= $1;
    } else {next;}

    if ($in[8] =~ $patpub || $in[8] =~ $patpub_unp){
        $pubid = $1;
    }else {next;}

        $method = $in[6];

        if ((exists $hcgs1{$id1}) && (exists $hcgs1{$id2})){
        $fb1 = $hcgs1{$id1};
        $fb2 = $hcgs1{$id2};
    }
    elsif((exists $hcgs1{$id1}) && (exists $hcgs2{$id2})){
```

```
            $fb1 = $hcgs1{$id1};
            $fb2 = $hcgs2{$id2};
        }
        elsif((exists $hcgs2{$id1}) && (exists $hcgs1{$id2})){
            $fb1 = $hcgs2{$id1};
            $fb2 = $hcgs1{$id2};
        }
        elsif((exists $hcgs2{$id1}) && (exists $hcgs2{$id2})){
            $fb1 = $hcgs2{$id1};
            $fb2 = $hcgs2{$id2};
        } else {next;}

        my $str_int = "$fb1\t$fb2\t$in[2]\t$in[3]\t$in[6]\t$pubid\tMINT-$dte\t$dte";
            print OTHER "$str_int\n";
            push @allint_arr , $str_int ;
}
#sort so that the data is consistent with how other interaction tables are built.
@allint_arr = sort(@allint_arr);

my $item;

foreach $item (@allint_arr){
        my @a = split (/\t/, $item);
        my $fb1= $a[0];
        my $fb2 = $a[1];
        my $pubid = $a[5];
        my $method = $a[4];
        my $db = $a[6];

        my $forw = "$fb1\t$fb2";
```

```
        my $rev = "$fb2\t$fb1";
        if ((exists $hint_db{$forw}) || (exists $hint_db{$rev})){
                if (exists $hint_db{$forw} ){ push @{ $hint_db{$forw} } , $db ; }
            elsif (exists $hint_db{$rev}){ push @{ $hint_db{$rev} } , $db ; }
        }
        else { $hint_db {$forw} = [ $db ] ; }
# pmid
        if ((exists $hint_pid{$forw}) || (exists $hint_pid{$rev})){
            if (exists $hint_pid{$forw} ){
                    push @{ $hint_pid{$forw} } , $pubid ;
                my $pforw ="$forw\t$pubid";
                if (exists $hint_met{$pforw} ){
                    push @{ $hint_met{$pforw} } , $method ;
                    #print "$pforw\n";
                }
                 }
             elsif (exists $hint_pid{$rev}) {
                 push @{ $hint_pid{$rev} } , $pubid ;
                 my $prev = "$rev\t$pubid";
                 if (exists $hint_met{$prev}) {
                         push @{ $hint_met{$prev} } , $method ;
                #        print "$prev\n";
                 }
                 else {$hint_met {$prev} = [ $method ] ; }
            }
            }
        else {
            $hint_pid {$forw} = [ $pubid ] ;
            my $pforw ="$forw\t$pubid";
            $hint_met {$pforw} = [ $method ] ;
```

```
            #print "$pforw\n";
        }
#method
        my $pforw ="$fb1\t$fb2\t$pubid";
        my $prev = "$fb2\t$fb1\t$pubid";
        if ((exists $hint_met{$pforw}) || (exists $hint_met{$prev})){
                if (exists $hint_met{$pforw} ){ push @{ $hint_met{$pforw} } , $method ; }
            elsif (exists $hint_met{$prev}) { push @{ $hint_met{$prev} } , $method ; }
        }
        else { $hint_met {$pforw} = [ $method ] ; }
}

my $ky;
foreach $ky (keys %hint_db){
            my @d = split (/\t/,  $ky);
            my $f1 = $d[0];
            my $f2 = $d[1];
            if ($f2 lt $f1){
                        print FINAL "$f2\t$f1\t";
            }
            else{
                        print FINAL "$ky\t";
            }
            my @dups_arr1 = @{ $hint_db{$ky} };
            my @no_dups1 = Remove_dups::rem_dups(@dups_arr1);
            my @dups_arr2 = @{ $hint_pid{$ky} };
            my @no_dups2 = Remove_dups::rem_dups(@dups_arr2);
            my $pid_str = join (",", @no_dups2);
            print FINAL "$pid_str\t";
            print FINAL "$url$pid_str&dopt=DocSum\t";
```

```
		my $x;
		my $met_str;
		foreach $x (@no_dups2) {
			$met_str .= $x;
#			print "$ky, $x\n";
			my $ky1 = "$ky\t$x";
			my @dups_arr3 = @{ $hint_met{$ky1} };
			my @no_dups3 = Remove_dups::rem_dups(@dups_arr3);
#			print join("-", @no_dups3), "\n";
			$met_str.= "(";
			$met_str .= join ("," , @no_dups3);
			$met_str .= "),";
		}
		$met_str = substr ($met_str, 0,length($met_str)-1);
	print FINAL "$met_str";
		print FINAL "\t", join (",", @no_dups1), "\t$dte\tpp\n";
}

close STDERR;
close OTHER;
close FINAL;
```

get_hs_interologs_vone.pl

```
#!usr/local/bin/perl
#
#:vim syntax=perl
#
#this script uses the psicquic tool to get data from Biogrid, intact and MINT using MIQL syntax
# read in fly FBgn, Gene Ids from output file of Inparanoid scripts and store as hash (inp_hs.txt)
```

```
# match IDs from the 5 dbs through intermediate files from ncbi and ensembl to FBgns and output the interologs with FBgns and scores etc.

# input file order entrez id file (Homo_sapiens.gene_info),

# uniprot mapping file from ensembl (mart_export_hs_uniprotkb.txt)

# and finally the human mapping output file from get_orthologs.pl

#

open (STDERR, ">error.out");

open (OTHER, ">hs_phy.txt");

open (FINAL, ">hs_interactions.txt");

open(INP, "<$ARGV[2]")  or die("Cannot open file '$ARGV[2]' for reading\n");

open(ENTR, "<$ARGV[0]")  or die("Cannot open file '$ARGV[0]' for reading\n");

open(UNIP, "<$ARGV[1]")  or die("Cannot open file '$ARGV[1]' for reading\n");

#open (OUT3, ">hs_.txt");

use strict;

use LWP::Simple;

use Thila::Remove_dups;


my %huni_ensg = ();

my %hhprd_ensg = ();

my %hentr_ensg = ();

my %hensg_fbgn = ();

my %hint_db = ();

my %hint_pid = ();

my %hint_met = ();

my @allint_arr;


my $url = "http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=retrieve&db=pubmed&list_uids=";

my $hupat = qr/Ensembl\:(ENSG\d+)/os;

my $hupat1 = qr/HPRD\:(\d+)/os;

my $hprdf = "BINARY_PROTEIN_PROTEIN_INTERACTIONS.txt";

my $reacf = "homo_sapiens.interactions.txt";
```

```
while (<ENTR>){
	chomp $_;
	my @line = split(/\t/, $_);
	my $eid = $line[1];
	my $pats = $line[5];
	my $hprd;
	my $ensg;
	if ($pats =~ $hupat){
		$ensg = $1;
	}
	if ($pats =~ $hupat1){
		$hprd = $1;
	}
	#this check is weird but may be necessary due to data quirks
	if (defined($hprd) && defined($ensg)){
		if ((exists $hhprd_ensg{$hprd}) && ( $hhprd_ensg{$hprd} ne $ensg)){
			$hhprd_ensg{$hprd} .= ", $ensg";
		}
		else { $hhprd_ensg{$hprd} =  $ensg;}
	}
	if (defined($ensg)){
		if ((exists $hentr_ensg{$eid}) && ( $hentr_ensg{$eid} ne $ensg)){
			$hentr_ensg{$eid} .= ", $ensg";
		}
		else { $hentr_ensg {$eid} =  $ensg;}
	}
}
close ENTR;
```

```
my $k1;
foreach $k1 (keys %hhprd_ensg){
	my $o = $hhprd_ensg{$k1};
	if ($o =~ /,/){ delete $hhprd_ensg{$k1};  }
}
my $k;
foreach $k (keys %hentr_ensg){
	my $o = $hentr_ensg{$k};
	if ($o =~ /,/){ delete $hentr_ensg{$k}; }
}

while (<UNIP>){
	chomp $_;
	if ($_ =~ /ENSG\d+/ ){
		my @line = split(/\t/, $_);
		my $ensg = $line[0];
		my $uni = $line[1];
		if (defined($uni)){
			if ((exists $huni_ensg{$uni}) && ( $huni_ensg{$uni} ne $ensg)){
			$huni_ensg{$uni} .= ", $ensg";
		}
		else { $huni_ensg {$uni} =  $ensg;}

		}
	}else {next;}
}
close UNIP;

my $i;
foreach $i (keys %huni_ensg){
```

```
        my $o = $huni_ensg{$i};
        if ($o =~ /,/){ delete $huni_ensg{$i};  }
}

while (<INP>){
        chomp $_;
        my @line = split (/\t/, $_);
        my $fb = $line[2];
        my $fp = $line[3];
        my $scr1 = $line[4];
        my $en = $line[5];
        my $enp = $line[6];
        my $scr2 = $line[7];
        my $h = "$en";
                my $scr = "$fb\t$scr1\t$scr2";
        if ((exists $hensg_fbgn{$h}) && ( $hensg_fbgn{$h} ne $scr)){
                $hensg_fbgn{$h} .= ",$scr";
        }
        else { $hensg_fbgn {$h} =  $scr;}
}
my $a;
my %hensg_fbgn1;
#foreach $a1 ( keys %hensg_fbgn) {
#       print OUT3 "$a1\t", join ("\t", $hensg_fbgn{$a1} ), "\n";
#}

foreach $a (keys %hensg_fbgn){
        my $o = $hensg_fbgn{$a};
        if ($o =~ /,/){
                        my @e = split (/,/, $o);
```

```
                my $i;
                foreach $i (@e){
                        my @ei = split (/\t/, $i);
                        my $r = "$a\t$ei[0]";
                        $hensg_fbgn1{$r} = "$ei[1]\t$ei[2]";
                }
                delete $hensg_fbgn{$a};
        }
}

close INP;

my $curr_day = (localtime)[3];
my @mon = ("JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", "DEC");
my $curr_mon = (localtime)[4];
my $curr_year = (localtime)[5]+1900;
my $dte= "$curr_day-";
$dte .=$mon[$curr_mon];
$dte .= "-";
$dte .= substr $curr_year, 2;
print "$dte -made hashes. Getting BioGrid data\n";

my $bl;
my $il;
my $ml;
my $content = ();
# biogrid get all physical interactions (MI OBO term 'association')
# AND both belonging to homo sapiens (9606)
```

```
$content = get("http://tyerslab.bio.ed.ac.uk:8080/psicquic-ws/webservices/current/search/query/type:%22*association%22%20AND%20taxidA:9606%20AND%20taxidB:9606");

die "Couldn't get it!" unless defined $content;

#print "$content\n";

my $patb = qr/entrez gene\/locuslink:(\d+)/os;

my $patpub = qr/pubmed:(\d+)/os;

my $patpub_unp = qr/pubmed:(\w+.*?)/os;


$content =~ s/"//g;

$content =~ s/psi-mi://g;


my @b = split (/\n/, $content);

#fields 3, 4 sym, 5, 6 CG, 7 meth, 9 pmid, add date download, db

foreach $bl (@b){

        my @in = split (/\t/, $bl);

        my $id1; my $id2;

        my $pubid;

        my $en1; my $en2;

        my $method;

        my @fb1; my @fb2;


        if ($in[0] =~ $patb) {

            $id1 = $1;

        }

        else {next;}

        if ($in[1] =~ $patb){

            $id2= $1;

        }

        else {next;}

        if ($in[8] =~ $patpub || $in[8] =~ $patpub_unp){
```

```
                $pubid = $1;
        } else {next;}
        $method = $in[6];

        if ((exists $hentr_ensg{$id1}) && (exists $hentr_ensg{$id2})){
                $en1 = $hentr_ensg{$id1};
                $en2 = $hentr_ensg{$id2};
    } else {next;}
        my $str_int;
        if ((exists $hensg_fbgn{$en1}) && (exists $hensg_fbgn{$en2})){
                @fb1 = split (/\t/, $hensg_fbgn{$en1});
                @fb2 = split (/\t/, $hensg_fbgn{$en2});
                my $scrf1 = "$fb1[1]\t$fb1[2]";
                my $scrf2 = "$fb2[1]\t$fb2[2]";
                my $fbgn1 = $fb1[0];
                my $fbgn2 = $fb2[0];
                if ($fbgn1 lt $fbgn2){
                        $str_int = "$fb1[0]\t$fb2[0]\t$en1\t$en2\t$scrf1\t$scrf2\t$method\t$pubid\tBioGRID-
$dte\t$dte";
                }
                else{
                        $str_int = "$fb2[0]\t$fb1[0]\t$en2\t$en1\t$scrf2\t$scrf1\t$method\t$pubid\tBioGRID-
$dte\t$dte";
                }
                print OTHER "$str_int\n";
                push @allint_arr , $str_int ;
    }
        elsif (exists $hensg_fbgn{$en1}){
                @fb1 = split (/\t/, $hensg_fbgn{$en1});
                my $fbgn1 = $fb1[0];
                my $scrf1 = "$fb1[1]\t$fb1[2]";
```

```
                my $ensg1 = $en1;
                my $kbi;
                foreach $kbi (keys %hensg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        my $ensg2 = $zi[0];
                        my $fbgn2 = $zi[1];
                        my $scrf2 = $hensg_fbgn1{$kbi};
                        if ($ensg2 eq $en2){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                }
                                print OTHER "$str_int\n";
                                push @allint_arr , $str_int ;
                        }
                }
        }
        elsif (exists $hensg_fbgn{$en2}){
                @fb2 = split (/\t/, $hensg_fbgn{$en2});
                my $ensg2 = $en2;
                my $fbgn2 = $fb2[0];
                my $scrf2 = "$fb2[1]\t$fb2[2]";
                my $kbi;
                foreach $kbi (keys %hensg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        my $ensg1 = $zi[0];
                        my $fbgn1 = $zi[1];
```

```
                        my $scrf1 = $hensg_fbgn1{$kbi};
                        if ($ensg1 eq $en1){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                }
                                print OTHER "$str_int\n";
                                push @allint_arr , $str_int ;
                        }
                }
        }
        else {
                my $kb;
                foreach $kb (keys %hensg_fbgn1){
                        my @z = split (/\t/, $kb);
                        my $ensg1 = $z[0];
                        my $fbgn1 = $z[1];
                        my $scrf1 = $hensg_fbgn1{$kb};
                        if ($ensg1 eq $en1){
                                my $scrf2;
                                my $kbi;
                                foreach $kbi (keys %hensg_fbgn1){
                                        my @zi = split (/\t/, $kbi);
                                        my $ensg2 = $zi[0];
                                        my $fbgn2 = $zi[1];
                                        $scrf2 = $hensg_fbgn1{$kbi};
                                        if ($ensg2 eq $en2){
```

```
                                                    if ($fbgn1 lt $fbgn2){

                                                            $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tBioGRID-$dte\t$dte";

                                                    }

                                                    else{

                                                            $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tBioGRID-$dte\t$dte";

                                                    }

                                                    print OTHER "$str_int\n";

                                                    push @allint_arr , $str_int ;

                                            }

                                    }

                            }

                    }

            }

}

#intact data

print "getting intact data\n";

$content = ();

$content =
get("http://www.ebi.ac.uk/Tools/webservices/psicquic/intact/webservices/current/search/query/type:%22*association%
22%20AND%20taxidA:9606%20AND%20taxidB:9606");

die "Couldn't get it!" unless defined $content;

#$content =~ s/uniprotkb://g;

$content =~ s/"//g;

$content =~ s/psi-mi://g;



#print "$content\n";

#my $pati0 = qr/ensembl\:(ENSG\d+)/os;

my $patid = qr/uniprotkb:(.*?)intact:EBI/os;

my @i = split (/\n/, $content);

foreach $il (@i) {
```

```
	my @in = split (/\t/, $il);
my $en1; my $en2;
my $pubid;
my $method;
	my @fb1; my @fb2;
	my $id1; my $id2;

	#if ($in[19] =~ $pati0) {
	#	$en1 = $1;
	#}else {next;}

	#if ($in[20] =~ $pati0){
	#	$en2= $1;
	#}else {next;}
	if ($in[0] =~ $patid){
			$id1= $1;
			chop $id1;
	}else {next;}
	if ($in[1] =~ $patid){
			$id2 = $1;
			chop $id2;
	}else {next;}

	if ((exists $huni_ensg{$id1}) && (exists $huni_ensg{$id2})){
	$en1 = $huni_ensg{$id1};
	$en2 = $huni_ensg{$id2};
} else {next;}

if ($in[8] =~ $patpub || $in[8] =~ $patpub_unp){
	$pubid = $1;
```

```
} else {next;}


        $method = $in[6];
        my $str_int;
        my $fbgn1; my $fbgn2;
        my $scrf1; my $scrf2;
        my $ensg1; my $ensg2;
        if ((exists $hensg_fbgn{$en1}) && (exists $hensg_fbgn{$en2})){
                @fb1 = split (/\t/, $hensg_fbgn{$en1});
                @fb2 = split (/\t/, $hensg_fbgn{$en2});
                $fbgn1 = $fb1[0];
                $fbgn2 = $fb2[0];
                $scrf1 = "$fb1[1]\t$fb1[2]";
                $scrf2 = "$fb2[1]\t$fb2[2]";
                if ($fbgn1 lt $fbgn2){
                        $str_int = "$fb1[0]\t$fb2[0]\t$en1\t$en2\t$scrf1\t$scrf2\t$method\t$pubid\tIntAct-
$dte\t$dte";
                }
                else{
                        $str_int = "$fb2[0]\t$fb1[0]\t$en2\t$en1\t$scrf2\t$scrf1\t$method\t$pubid\tIntAct-
$dte\t$dte";
                }
                print OTHER "$str_int\n";
                push @allint_arr , $str_int ;
        }
        elsif (exists $hensg_fbgn{$en1}){
                @fb1 = split (/\t/, $hensg_fbgn{$en1});
                $fbgn1 = $fb1[0];
                $ensg1 = $en1;
                $scrf1 = "$fb1[1]\t$fb1[2]";
                my $kbi;
```

```
                foreach $kbi (keys %hensg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        $ensg2 = $zi[0];
                        $fbgn2 = $zi[1];
                        $scrf2 = $hensg_fbgn1{$kbi};
                        if ($ensg2 eq $en2){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tIntAct-$dte\t$dte";
                                }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tIntAct-$dte\t$dte";
                                }
                                print OTHER "$str_int\n";
                                push @allint_arr , $str_int ;
                        }
                }
        }
        elsif (exists $hensg_fbgn{$en2}){
                @fb2 = split (/\t/, $hensg_fbgn{$en2});
                $ensg2 = $en2;
                $fbgn2 = $fb2[0];
                $scrf2 = "$fb2[1]\t$fb2[2]";
                my $kbi;
                foreach $kbi (keys %hensg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        $ensg1 = $zi[0];
                        $fbgn1 = $zi[1];
                        $scrf1 = $hensg_fbgn1{$kbi};
                        if ($ensg1 eq $en1){
```

```
                                        if ($fbgn1 lt $fbgn2){

                                                $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tIntAct-$dte\t$dte";

                                        }

                                        else{

                                                $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tIntAct-$dte\t$dte";

                                        }

                                        print OTHER "$str_int\n";

                                        push @allint_arr , $str_int ;

                                }

                        }

                }

                else{

                        my $kb;

                        foreach $kb (keys %hensg_fbgn1){

                                my @z = split (/\t/, $kb);

                                $ensg1 = $z[0];

                                $fbgn1 = $z[1];

                                $scrf1 = $hensg_fbgn1{$kb};

                                if ($ensg1 eq $en1){

                                        my $scrf2;

                                        my $kbi;

                                        foreach $kbi (keys %hensg_fbgn1){

                                                my @zi = split (/\t/, $kbi);

                                                my $ensg2 = $zi[0];

                                                my $fbgn2 = $zi[1];

                                                $scrf2 = $hensg_fbgn1{$kbi};

                                                if ($ensg2 eq $en2){


                                                        if ($fbgn1 lt $fbgn2){
```

```
                                                                $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tIntAct-$dte\t$dte";

                                                        }

                                                        else{

                                                                $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tIntAct-$dte\t$dte";

                                                        }

                                                        print OTHER "$str_int\n";

                                                        push @allint_arr , $str_int ;

                                                }

                                        }

                                }

                        }

                }


}


# MINT - gets all interaction data - type doesn't work - so need to make sure to only get 'association' type

print "getting MINT data\n";

$content = ();

$content =
get("http://mint.bio.uniroma2.it/mint/psicquic/webservices/current/search/query/taxidA:9606%20AND%20taxidB:960
6");

die "Couldn't get it!" unless defined $content;


#print "$content\n";

$content =~ s/uniprotkb://g;


$content =~ s/"//g;

$content =~ s/psi-mi://g;


my @m = split (/\n/, $content);
```

```
my $rem_pat = qr/MI:0403\(colocalization\)/os; #field 12

# MINT doesn't have genetic interactions - so need to only remove colocalization
# fields 5, 6 (CG), 7 (meth), 9 pmid, date, db
foreach $ml (@m) {
        if ($ml =~ $rem_pat) {next;}
        my @in = split (/\t/, $ml);
    my $id1; my $id2;
    my $pubid;
    my $en1; my $en2;
    my $method;
        my @fb1; my @fb2;

    $id1 = $in[0];

    $id2 =$in[1];
#       print "$id1\t$id2\n";
    if ($in[8] =~ $patpub || $in[8] =~ $patpub_unp){
        $pubid = $1;
    }else {next;}

        $method = $in[6];

        if ((exists $huni_ensg{$id1}) && (exists $huni_ensg{$id2})){
        $en1 = $huni_ensg{$id1};
        $en2 = $huni_ensg{$id2};
    } else {next;}
        my $str_int;
        my $fbgn1; my $fbgn2;
        my $scrf1; my $scrf2;
```

```
		my $ensg1; my $ensg2;
		if ((exists $hensg_fbgn{$en1}) && (exists $hensg_fbgn{$en2})){
			@fb1 = split (/\t/, $hensg_fbgn{$en1});
			@fb2 = split (/\t/, $hensg_fbgn{$en2});
			$scrf1 = "$fb1[1]\t$fb1[2]";
			$scrf2 = "$fb2[1]\t$fb2[2]";
			$fbgn1 = $fb1[0];
			$fbgn2 = $fb2[0];
			if ($fbgn1 lt $fbgn2){
				$str_int = "$fb1[0]\t$fb2[0]\t$en1\t$en2\t$scrf1\t$scrf2\t$method\t$pubid\tMINT-
$dte\t$dte";
			}
			else{
				$str_int = "$fb2[0]\t$fb1[0]\t$en2\t$en1\t$scrf2\t$scrf1\t$method\t$pubid\tMINT-
$dte\t$dte";
			}
			print OTHER "$str_int\n";
			push @allint_arr , $str_int ;
	}
		elsif (exists $hensg_fbgn{$en1}){
			@fb1 = split (/\t/, $hensg_fbgn{$en1});
			$fbgn1 = $fb1[0];
			$ensg1 = $en1;
			$scrf1 = "$fb1[1]\t$fb1[2]";
			my $kbi;
			foreach $kbi (keys %hensg_fbgn1){
				my @zi = split (/\t/, $kbi);
				$ensg2 = $zi[0];
				$fbgn2 = $zi[1];
				$scrf2 = $hensg_fbgn1{$kbi};
				if ($ensg2 eq $en2){
```

```
                                        if ($fbgn1 lt $fbgn2){
                                                $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tMINT-$dte\t$dte";
                                        }
                                        else{
                                                $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tMINT-$dte\t$dte";
                                        }
                                        print OTHER "$str_int\n";
                                        push @allint_arr , $str_int ;
                                }
                        }
                }
                elsif (exists $hensg_fbgn{$en2}){
                        @fb2 = split (/\t/, $hensg_fbgn{$en2});
                        $ensg2 = $en2;
                        $fbgn2 = $fb2[0];
                        $scrf2 = "$fb2[1]\t$fb2[2]";
                        my $kbi;
                        foreach $kbi (keys %hensg_fbgn1){
                                my @zi = split (/\t/, $kbi);
                                $ensg1 = $zi[0];
                                $fbgn1 = $zi[1];
                                $scrf1 = $hensg_fbgn1{$kbi};
                                if ($ensg1 eq $en1){
                                        if ($fbgn1 lt $fbgn2){
                                                $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tMINT-$dte\t$dte";
                                        }
                                        else{
                                                $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tMINT-$dte\t$dte";
```

```
                                        }
                                        print OTHER "$str_int\n";
                                        push @allint_arr , $str_int ;
                                }
                        }
                }
                else{
                        my $kb;
                        foreach $kb (keys %hensg_fbgn1){
                                my @z = split (/\t/, $kb);
                                $ensg1 = $z[0];
                                $fbgn1 = $z[1];
                                $scrf1 = $hensg_fbgn1{$kb};
                                if ($ensg1 eq $en1){
                                        my $kbi;
                                        foreach $kbi (keys %hensg_fbgn1){
                                                my @zi = split (/\t/, $kbi);
                                                $ensg2 = $zi[0];
                                                $fbgn2 = $zi[1];
                                                $scrf2 = $hensg_fbgn1{$kbi};
                                                if ($ensg2 eq $en2){

                                                        if ($fbgn1 lt $fbgn2){
                                                                $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tMINT-$dte\t$dte";
                                                        }
                                                        else{
                                                                $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tMINT-$dte\t$dte";
                                                        }
                                                        print OTHER "$str_int\n";
```

```
                                                push @allint_arr , $str_int ;
                                        }
                                }
                        }
                }
        }
}
print "getting hprd data\n";
open(HPR, "<$hprdf")  or die("Cannot open file '$hprdf' for reading\n");
while (<HPR>){
        chomp $_;
        my @in = split (/\t/, $_);
        my $id1; my $id2;
        $id1 = $in[1];
        $id2 = $in[4];
        my $method;
    if (defined( $in[6])){
                $method = $in[6];
        }
        else{
                $method = 'NA';
        }
        my $pubid;
    if (defined ($in[7])){
                $pubid = $in[7];
        }
        else {
                $pubid = 'NA';
        }
        my @fb1; my @fb2;
```

```
        my $en1; my $en2;


        if ((exists $hhprd_ensg{$id1}) && (exists $hhprd_ensg{$id2})){
        $en1 = $hhprd_ensg{$id1};
        $en2 = $hhprd_ensg{$id2};
    } else {next;}
        my $str_int;
        my $fbgn1; my $fbgn2;
        my $scrf1; my $scrf2;
        my $ensg1; my $ensg2;
        if ((exists $hensg_fbgn{$en1}) && (exists $hensg_fbgn{$en2})){
                @fb1 = split (/\t/, $hensg_fbgn{$en1});
                @fb2 = split (/\t/, $hensg_fbgn{$en2});
                $scrf1 = "$fb1[1]\t$fb1[2]";
                $scrf2 = "$fb2[1]\t$fb2[2]";
                $fbgn1 = $fb1[0];
                $fbgn2 = $fb2[0];
                if ($fbgn1 lt $fbgn2){
                        $str_int = "$fb1[0]\t$fb2[0]\t$en1\t$en2\t$scrf1\t$scrf2\t$method\t$pubid\tHPRD-
$dte\t$dte";
                }
                else{
                        $str_int = "$fb2[0]\t$fb1[0]\t$en2\t$en1\t$scrf2\t$scrf1\t$method\t$pubid\tHPRD-
$dte\t$dte";
                }
                print OTHER "$str_int\n";
                push @allint_arr , $str_int ;
    }
        elsif (exists $hensg_fbgn{$en1}){
                @fb1 = split (/\t/, $hensg_fbgn{$en1});
                $fbgn1 = $fb1[0];
```

```
                $ensg1 = $en1;
                $scrf1 = "$fb1[1]\t$fb1[2]";
                my $kbi;
                foreach $kbi (keys %hensg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        $ensg2 = $zi[0];
                        $fbgn2 = $zi[1];
                        $scrf2 = $hensg_fbgn1{$kbi};
                        if ($ensg2 eq $en2){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tHPRD-$dte\t$dte";
                                }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tHPRD-$dte\t$dte";
                                }
                                print OTHER "$str_int\n";
                                push @allint_arr , $str_int ;
                        }
                }
        }
        elsif (exists $hensg_fbgn{$en2}){
                @fb2 = split (/\t/, $hensg_fbgn{$en2});
                $ensg2 = $en2;
                $fbgn2 = $fb2[0];
                $scrf2 = "$fb2[1]\t$fb2[2]";
                my $kbi;
                foreach $kbi (keys %hensg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        $ensg1 = $zi[0];
```

```
                        $fbgn1 = $zi[1];
                        $scrf1 = $hensg_fbgn1{$kbi};
                        if ($ensg1 eq $en1){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tHPRD-$dte\t$dte";
                                }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tHPRD-$dte\t$dte";
                                }
                                print OTHER "$str_int\n";
                                push @allint_arr , $str_int ;
                        }
                }
        }
        else{
                my $kb;
                foreach $kb (keys %hensg_fbgn1){
                        my @z = split (/\t/, $kb);
                        $ensg1 = $z[0];
                        $fbgn1 = $z[1];
                        $scrf1 = $hensg_fbgn1{$kb};
                        if ($ensg1 eq $en1){
                                my $kbi;
                                foreach $kbi (keys %hensg_fbgn1){
                                        my @zi = split (/\t/, $kbi);
                                        $ensg2 = $zi[0];
                                        $fbgn2 = $zi[1];
                                        $scrf2 = $hensg_fbgn1{$kbi};
                                        if ($ensg2 eq $en2){
```

```
                                                    if ($fbgn1 lt $fbgn2){
                                                            $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tHPRD-$dte\t$dte";
                                                    }
                                                    else{
                                                            $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tHPRD-$dte\t$dte";
                                                    }
                                                    print OTHER "$str_int\n";
                                                    push @allint_arr , $str_int ;
                                            }
                                    }
                            }
                    }
            }
}
close HPR;

print "getting Reactome data\n";
open(REA, "<$reacf")  or die("Cannot open file '$reacf' for reading\n");
my %hraw;
while (<REA>){
            chomp $_;
            my $l = $_;
            $l =~ s/UniProt://g;
            my @in = split (/\t/, $l);
            my $patr = qr/ENSEMBL:(ENSG\d+)/os;
#           my $patr0 = qr//os;
            my $en1; my $en2;
            my $id1 = $in[0];
```

```
	my $id2 = $in[3];
	my @fb1;
	my @fb2;

	if (($in[1] =~ $patr) && ($in[2] =~ $patr)){
		if ($in[1] =~ $patr){
			$en1 = $1;
		}
		if ($in[4] =~ $patr) {
			$en2 = $1;
		}
	}
	elsif ((exists $huni_ensg{$id1}) && (exists $huni_ensg{$id2})){
			$en1 = $huni_ensg{$id1};
			$en2 = $huni_ensg{$id2};
	} else {next;}

	my $pubid;
	if (defined  ($in[8])){
		$pubid = $in[8];
	}
	else {
		$pubid = 'NA';
	}
	my $method;
	if (defined ($in[6])){
	    $method = $in[6];
}
 else{
		$method = 'NA';
```

```
        }
        my $needed = "$en1\t$en2\t$pubid\t$method";
        if (exists $hraw{$needed}){next;}
        else{ $hraw{$needed} = 'first';}

        my $str_int;
        my $fbgn1; my $fbgn2;
        my $scrf1; my $scrf2;
        my $ensg1; my $ensg2;
        if ((exists $hensg_fbgn{$en1}) && (exists $hensg_fbgn{$en2})){
                @fb1 = split (/\t/, $hensg_fbgn{$en1});
                @fb2 = split (/\t/, $hensg_fbgn{$en2});
                $scrf1 = "$fb1[1]\t$fb1[2]";
                $scrf2 = "$fb2[1]\t$fb2[2]";
                $fbgn1 = $fb1[0];
                $fbgn2 = $fb2[0];
                if ($fbgn1 lt $fbgn2){
                        $str_int = "$fb1[0]\t$fb2[0]\t$en1\t$en2\t$scrf1\t$scrf2\t$method\t$pubid\tREACTOME-
$dte\t$dte";
                }
                else{
                        $str_int = "$fb2[0]\t$fb1[0]\t$en2\t$en1\t$scrf2\t$scrf1\t$method\t$pubid\tREACTOME-
$dte\t$dte";
                }
                print OTHER "$str_int\n";
                push @allint_arr , $str_int ;
    }
        elsif (exists $hensg_fbgn{$en1}){
                @fb1 = split (/\t/, $hensg_fbgn{$en1});
                $fbgn1 = $fb1[0];
                $ensg1 = $en1;
```

```
                $scrf1 = "$fb1[1]\t$fb1[2]";
                my $kbi;
                foreach $kbi (keys %hensg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        $ensg2 = $zi[0];
                        $fbgn2 = $zi[1];
                        $scrf2 = $hensg_fbgn1{$kbi};
                        if ($ensg2 eq $en2){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tREACTOME-$dte\t$dte";
                                }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tREACTOME-$dte\t$dte";
                                }
                                print OTHER "$str_int\n";
                                push @allint_arr , $str_int ;
                        }
                }
        }
        elsif (exists $hensg_fbgn{$en2}){
                @fb2 = split (/\t/, $hensg_fbgn{$en2});
                $ensg2 = $en2;
                $fbgn2 = $fb2[0];
                $scrf2 = "$fb2[1]\t$fb2[2]";
                my $kbi;
                foreach $kbi (keys %hensg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        $ensg1 = $zi[0];
                        $fbgn1 = $zi[1];
```

```
                        $scrf1 = $hensg_fbgn1{$kbi};
                        if ($ensg1 eq $en1){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tREACTOME-$dte\t$dte";
                        }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tREACTOME-$dte\t$dte";
                                }
                                print OTHER "$str_int\n";
                                push @allint_arr , $str_int ;
                        }
                }
        }
        else{
                my $kb;
                foreach $kb (keys %hensg_fbgn1){
                        my @z = split (/\t/, $kb);
                        $ensg1 = $z[0];
                        $fbgn1 = $z[1];
                        $scrf1 = $hensg_fbgn1{$kb};
                        if ($ensg1 eq $en1){
                                my $kbi;
                                foreach $kbi (keys %hensg_fbgn1){
                                        my @zi = split (/\t/, $kbi);
                                        $ensg2 = $zi[0];
                                        $fbgn2 = $zi[1];
                                        $scrf2 = $hensg_fbgn1{$kbi};
                                        if ($ensg2 eq $en2){
```

```
                                                    if ($fbgn1 lt $fbgn2){

                                                            $str_int =
"$fbgn1\t$fbgn2\t$ensg1\t$ensg2\t$scrf1\t$scrf2\t$method\t$pubid\tREACTOME-$dte\t$dte";

                                                    }

                                                    else{

                                                            $str_int =
"$fbgn2\t$fbgn1\t$ensg2\t$ensg1\t$scrf2\t$scrf1\t$method\t$pubid\tREACTOME-$dte\t$dte";

                                                    }

                                                    print OTHER "$str_int\n";

                                                    push @allint_arr , $str_int ;

                                            }

                                    }

                            }

                    }

            }


}


close REA;


print "building table\n\n\n";

#sort so that the data is consistent with how other interaction tables are built.

@allint_arr = sort(@allint_arr);


my $item;

my %hint_ens1;

my %hint_ens2;

my %hint_fbs;


foreach $item (@allint_arr){

        my @a = split (/\t/, $item);
```

```
	my $fb1= $a[0];
	my $fb2 = $a[1];
		my $en1 = $a[2];
		my $en2 = $a[3];
		my $sf1 = $a[4];
		my $se1 = $a[5];
		my $sf2 = $a[6];
		my $se2 = $a[7];
	my $pubid = $a[9];
	my $method = $a[8];
	my $db = $a[10];
		my $ens1 = "$en1,$se1";
		my $ens2 = "$en2,$se2";
		my $fbs = "$sf1,$sf2";

	my $forw = "$fb1\t$fb2";
	if (exists $hint_db{$forw}) { push @{ $hint_db{$forw} } , $db ; }
	else { $hint_db {$forw} = [ $db ] ; }

	if (exists $hint_fbs{$forw}) { push @{ $hint_fbs{$forw} } , $fbs ; }
	else { $hint_fbs {$forw} = [ $fbs ] ; }

		if (exists $hint_ens1{$forw}) { push @{ $hint_ens1{$forw} } , $ens1 ; }
	else { $hint_ens1{$forw} = [ $ens1 ] ; }

		if (exists $hint_ens2{$forw}) { push @{ $hint_ens2{$forw} } , $ens2 ; }
	else { $hint_ens2{$forw} = [ $ens2 ] ; }

# pmid
	if (exists $hint_pid{$forw}){
```

```
                push @{ $hint_pid{$forw} } , $pubid ;
                my $pforw ="$forw\t$pubid";
                if (exists $hint_met{$pforw} ){
                        push @{ $hint_met{$pforw} } , $method ;
                #print "$pforw\n";
                }
        }
        else {
                $hint_pid {$forw} = [ $pubid ] ;
                my $pforw ="$forw\t$pubid";
                $hint_met {$pforw} = [ $method ] ;
                #print "$pforw\n";
        }
#method
        my $pforw ="$fb1\t$fb2\t$pubid";
        if (exists $hint_met{$pforw}){ push @{ $hint_met{$pforw} } , $method ; }
        else { $hint_met {$pforw} = [ $method ] ; }
}

my $ky;
foreach $ky (keys %hint_db){
                print FINAL "$ky\t";
                my @dups_arr1 = @{ $hint_db{$ky} };
                my @no_dups1 = Remove_dups::rem_dups(@dups_arr1);

                my @dups_fs = @{ $hint_fbs{$ky} };
                my @no_dupsfs = Remove_dups::rem_dups(@dups_fs);

                my @dups_es1 = @{ $hint_ens1{$ky} };
                my @no_dupes1 = Remove_dups::rem_dups(@dups_es1);
```

```
my @dups_ens2 = @{ $hint_ens2{$ky} };
my @no_dupes2 = Remove_dups::rem_dups(@dups_ens2);

my @dups_arr2 = @{ $hint_pid{$ky} };
my @no_dups2 = Remove_dups::rem_dups(@dups_arr2);
my $pid_str = join (",", @no_dups2);
my @dups2_1 = split (/,/, $pid_str);
my @no_dups2_1 = Remove_dups::rem_dups(@dups2_1);
$pid_str = join (",", @no_dups2_1);
print FINAL "$pid_str\t";
print FINAL "$url$pid_str&dopt=DocSum\t";

my $f; #fbgn scores
my @f1; my @f2;
foreach $f (@no_dupsfs){
        my @s = split (/,/, $f);
        push @f1, $s[0];
        push @f2, $s[1];
}

my $e; # ensg1 and score
my @e1; my @e2;
foreach $e (@no_dupes1){
        my @s = split (/,/, $e);
        push @e1, $s[0];
        push @e2, $s[1];
}

my $d; #ensg2 and score
```

```
my @d1; my @d2;
foreach $d (@no_dupes2){
        my @s = split (/,/, $d);
        push @d1, $s[0];
        push @d2, $s[1];
}

my $str_scr = join (",", @f1);
$str_scr.= "\t";
$str_scr .= join (",", @f2);
$str_scr .= "\t";
$str_scr .= join (",", @e1);
$str_scr .= "\t";
$str_scr .= join (",", @e2);
$str_scr .= "\t";
$str_scr .= join (",", @d1);
$str_scr .= "\t";
$str_scr .= join (",", @d2);

print FINAL "$str_scr\t";

my $x;
my $met_str;
foreach $x (@no_dups2) {
        $met_str .= $x;
#       print "$ky, $x\n";
        my $ky1 = "$ky\t$x";
        my @dups_arr3 = @{ $hint_met{$ky1} };
        my @no_dups3 = Remove_dups::rem_dups(@dups_arr3);
#       print join("-", @no_dups3), "\n";
```

```
                $met_str.= "(";
                $met_str .= join ("," , @no_dups3);
                $met_str .= "),";
        }
        $met_str = substr ($met_str, 0,length($met_str)-1);
    print FINAL "$met_str";
        print FINAL "\t", join (",", @no_dups1), "\t$dte\tpp\n";
}

close STDERR;
close OTHER;
close FINAL;
```

get_sc_interologs.pl

```
#!usr/local/bin/perl
#
#:vim syntax=perl
#
# this script uses the psicquic tool to get Yeast data from Biogrid, intact and MINT using MIQL syntax
# read in fly FBgn, Gene Ids from output file of Inparanoid scripts and store as hash
# match IDs from these dbs through intermediate files from ncbi and ensembl to FBgns and output the interologs with
FBgns and scores etc.
# input file order entrez id file and the mapping output file from get_orthologs.pl

open (STDERR, ">error.out");
open (OTHER, ">sc_phy.txt");
open (FINAL, ">sc_interactions.txt");
open(INP, "<$ARGV[1]")  or die("Cannot open file '$ARGV[1]' for reading\n");
open(ENTR, "<$ARGV[0]")  or die("Cannot open file '$ARGV[0]' for reading\n");
use strict;
use LWP::Simple;
use Thila::Remove_dups;

my %hentr_sg = ();
my %hsg_fbgn = ();
my %hint_db = ();
my %hint_pid = ();
my %hint_met = ();
my @allint_arr;

my $url = "http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=retrieve&db=pubmed&list_uids=";
#my $scpat = qr/([Y|Q].*?)/os;

while (<ENTR>){
    chomp $_;
```

```
	my @line = split(/\t/, $_);
	my $eid = $line[1];
 #	my $pats = $line[5];
	my $sg = $line[3];
#	if ($pats =~ $scpat){
#		$sg = $1;
#	}
	#this check is weird but may be necessary due to data quirks
	if ( defined($sg)){
		if ((exists $hentr_sg{$eid}) && ( $hentr_sg{$eid} ne $sg)){
			$hentr_sg{$eid} .= ", $sg";
		}
		else { $hentr_sg {$eid} =  $sg;}
	}
}
close ENTR;

my $k;
foreach $k (keys %hentr_sg){
	my $o = $hentr_sg{$k};
	if ($o =~ /,/){ delete $hentr_sg{$k}; }
}

while (<INP>){
	chomp $_;
	my @line = split (/\t/, $_);
	my $fb = $line[2];
	my $fp = $line[3];
	my $scr1 = $line[4];
	my $en = $line[5];
	my $enp = $line[6];
	my $scr2 = $line[7];
	my $h = "$en";
		my $scr = "$fb\t$scr1\t$scr2";
	if ((exists $hsg_fbgn{$h}) && ( $hsg_fbgn{$h} ne $scr)){
		$hsg_fbgn{$h} .= ",$scr";
	}
	else { $hsg_fbgn {$h} =  $scr;}
}
my $a;
my %hsg_fbgn1;

foreach $a (keys %hsg_fbgn){
	my $o = $hsg_fbgn{$a};
	if ($o =~ /,/){
				my @e = split (/,/, $o);
				my $i;
				foreach $i (@e){
					my @ei = split (/\t/, $i);
					my $r = "$a\t$ei[0]";
					$hsg_fbgn1{$r} = "$ei[1]\t$ei[2]";
				}
				delete $hsg_fbgn{$a};
		}
}

close INP;

my $curr_day = (localtime)[3];
```

```
my @mon = ("JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", "DEC");
my $curr_mon = (localtime)[4];
my $curr_year = (localtime)[5]+1900;
my $dte= "$curr_day-";
$dte .=$mon[$curr_mon];
$dte .= "-";
$dte .= substr $curr_year, 2;
print "$dte -made hashes. Getting BioGrid data\n";

my $bl;
my $il;
my $ml;
my $content = ();
# biogrid get all physical interactions (MI OBO term 'association')
# AND both belonging to yeast

$content = get("http://tyerslab.bio.ed.ac.uk:8080/psicquic-
ws/webservices/current/search/query/type:%22*association%22%20AND%20taxidA:4932%20AND%20taxidB:4932"
);
die "Couldn't get it!" unless defined $content;
#print "$content\n";
my $patb = qr/entrez gene\/locuslink:(\d+)/os;
my $patpub = qr/pubmed:(\d+)/os;
my $patpub_unp = qr/pubmed:(\w+.*?)/os;

$content =~ s/"//g;
$content =~ s/psi-mi://g;

my @b = split (/\n/, $content);
#fields 3, 4 sym, 5, 6 CG, 7 meth, 9 pmid, add date download, db
foreach $bl (@b){
        my @in = split (/\t/, $bl);
        my $id1; my $id2;
        my $pubid;
        my $en1; my $en2;
        my $method;
        my @fb1; my @fb2;

        if ($in[0] =~ $patb) {
            $id1 = $1;
        }
        else {next;}
        if ($in[1] =~ $patb){
            $id2= $1;
        }
        else {next;}
        if ($in[8] =~ $patpub || $in[8] =~ $patpub_unp){
                $pubid = $1;
        } else {next;}
        $method = $in[6];

        if ((exists $hentr_sg{$id1}) && (exists $hentr_sg{$id2})){
                $en1 = $hentr_sg{$id1};
                $en2 = $hentr_sg{$id2};
    } else {next;}
        my $str_int;
        if ((exists $hsg_fbgn{$en1}) && (exists $hsg_fbgn{$en2})){
                @fb1 = split (/\t/, $hsg_fbgn{$en1});
                @fb2 = split (/\t/, $hsg_fbgn{$en2});
```

```
                my $scrf1 = "$fb1[1]\t$fb1[2]";
                my $scrf2 = "$fb2[1]\t$fb2[2]";
                my $fbgn1 = $fb1[0];
                my $fbgn2 = $fb2[0];
                if ($fbgn1 lt $fbgn2){
                        $str_int = "$fb1[0]\t$fb2[0]\t$en1\t$en2\t$scrf1\t$scrf2\t$method\t$pubid\tBioGRID-
$dte\t$dte";
                }
                else{
                        $str_int = "$fb2[0]\t$fb1[0]\t$en2\t$en1\t$scrf2\t$scrf1\t$method\t$pubid\tBioGRID-
$dte\t$dte";
                }
                print OTHER "$str_int\n";
                push @allint_arr , $str_int ;
        }
        elsif (exists $hsg_fbgn{$en1}){
                @fb1 = split (/\t/, $hsg_fbgn{$en1});
                my $fbgn1 = $fb1[0];
                my $scrf1 = "$fb1[1]\t$fb1[2]";
                my $sg1 = $en1;
                my $kbi;
                foreach $kbi (keys %hsg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        my $sg2 = $zi[0];
                        my $fbgn2 = $zi[1];
                        my $scrf2 = $hsg_fbgn1{$kbi};
                        if ($sg2 eq $en2){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$sg1\t$sg2\t$scrf1\t$scrf2\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$sg2\t$sg1\t$scrf2\t$scrf1\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                }
                                print OTHER "$str_int\n";
                                push @allint_arr , $str_int ;
                        }
                }
        }
        elsif (exists $hsg_fbgn{$en2}){
                @fb2 = split (/\t/, $hsg_fbgn{$en2});
                my $sg2 = $en2;
                my $fbgn2 = $fb2[0];
                my $scrf2 = "$fb2[1]\t$fb2[2]";
                my $kbi;
                foreach $kbi (keys %hsg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        my $sg1 = $zi[0];
                        my $fbgn1 = $zi[1];
                        my $scrf1 = $hsg_fbgn1{$kbi};
                        if ($sg1 eq $en1){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$sg1\t$sg2\t$scrf1\t$scrf2\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$sg2\t$sg1\t$scrf2\t$scrf1\t$method\t$pubid\tBioGRID-$dte\t$dte";
```

```
                                        }
                                        print OTHER "$str_int\n";
                                        push @allint_arr , $str_int ;
                                }
                        }
                }
                else {
                        my $kb;
                        foreach $kb (keys %hsg_fbgn1){
                                my @z = split (/\t/, $kb);
                                my $sg1 = $z[0];
                                my $fbgn1 = $z[1];
                                my $scrf1 = $hsg_fbgn1{$kb};
                                if ($sg1 eq $en1){
                                        my $scrf2;
                                        my $kbi;
                                        foreach $kbi (keys %hsg_fbgn1){
                                                my @zi = split (/\t/, $kbi);
                                                my $sg2 = $zi[0];
                                                my $fbgn2 = $zi[1];
                                                $scrf2 = $hsg_fbgn1{$kbi};
                                                if ($sg2 eq $en2){
                                                        if ($fbgn1 lt $fbgn2){
                                                                $str_int =
"$fbgn1\t$fbgn2\t$sg1\t$sg2\t$scrf1\t$scrf2\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                                        }
                                                        else{
                                                                $str_int =
"$fbgn2\t$fbgn1\t$sg2\t$sg1\t$scrf2\t$scrf1\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                                        }
                                                        print OTHER "$str_int\n";
                                                        push @allint_arr , $str_int ;
                                                }
                                        }
                                }
                        }
                }
}
#intact data
print "getting intact data\n";
$content = ();
$content =
get("http://www.ebi.ac.uk/Tools/webservices/psicquic/intact/webservices/current/search/query/type:%22*association%
22%20AND%20taxidA:4932%20AND%20taxidB:4932");
die "Couldn't get it!" unless defined $content;
#$content =~ s/uniprotkb://g;
$content =~ s/"//g;
$content =~ s/psi-mi://g;

#print "$content\n";
#my $pati0 = qr/ensembl\:([Y|Q].*?)\|/os; # changed 24th Jan 2011
my $pati0 = qr/uniprotkb\:([Y|Q].*?)\(locus name\)/os;
my @i = split (/\n/, $content);
foreach $il (@i) {
        my @in = split (/\t/, $il);
    my $en1; my $en2;
    my $pubid;
    my $method;
        my @fb1; my @fb2;
```

```
	if ($in[2] =~ $pati0) {
	    $en1 = $1;
	      }else {next;}
#	      print "$en1\t";
	if ($in[3] =~ $pati0){
	    $en2= $1;
	      }else {next;}
#	      print "$en2\n";

	if ($in[8] =~ $patpub || $in[8] =~ $patpub_unp){
	      $pubid = $1;
	} else {next;}

	      $method = $in[6];
	      my $str_int;
	      my $fbgn1; my $fbgn2;
	      my $scrf1; my $scrf2;
	      my $sg1; my $sg2;
	      if ((exists $hsg_fbgn{$en1}) && (exists $hsg_fbgn{$en2})){
			@fb1 = split (/\t/, $hsg_fbgn{$en1});
			@fb2 = split (/\t/, $hsg_fbgn{$en2});
			$fbgn1 = $fb1[0];
			$fbgn2 = $fb2[0];
			$scrf1 = "$fb1[1]\t$fb1[2]";
			$scrf2 = "$fb2[1]\t$fb2[2]";
			if ($fbgn1 lt $fbgn2){
				$str_int = "$fb1[0]\t$fb2[0]\t$en1\t$en2\t$scrf1\t$scrf2\t$method\t$pubid\tIntAct-
$dte\t$dte";
			}
			else{
				$str_int = "$fb2[0]\t$fb1[0]\t$en2\t$en1\t$scrf2\t$scrf1\t$method\t$pubid\tIntAct-
$dte\t$dte";
			}
			print OTHER "$str_int\n";
			push @allint_arr , $str_int ;
	}
	      elsif (exists $hsg_fbgn{$en1}){
			@fb1 = split (/\t/, $hsg_fbgn{$en1});
			$fbgn1 = $fb1[0];
			$sg1 = $en1;
			$scrf1 = "$fb1[1]\t$fb1[2]";
			my $kbi;
			foreach $kbi (keys %hsg_fbgn1){
				my @zi = split (/\t/, $kbi);
				$sg2 = $zi[0];
				$fbgn2 = $zi[1];
				$scrf2 = $hsg_fbgn1{$kbi};
				if ($sg2 eq $en2){
					if ($fbgn1 lt $fbgn2){
						$str_int =
"$fbgn1\t$fbgn2\t$sg1\t$sg2\t$scrf1\t$scrf2\t$method\t$pubid\tIntAct-$dte\t$dte";
					}
					else{
						$str_int =
"$fbgn2\t$fbgn1\t$sg2\t$sg1\t$scrf2\t$scrf1\t$method\t$pubid\tIntAct-$dte\t$dte";
					}
					print OTHER "$str_int\n";
					push @allint_arr , $str_int ;
```

```
                        }
                }
        }
        elsif (exists $hsg_fbgn{$en2}){
                @fb2 = split (/\t/, $hsg_fbgn{$en2});
                $sg2 = $en2;
                $fbgn2 = $fb2[0];
                $scrf2 = "$fb2[1]\t$fb2[2]";
                my $kbi;
                foreach $kbi (keys %hsg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        $sg1 = $zi[0];
                        $fbgn1 = $zi[1];
                        $scrf1 = $hsg_fbgn1{$kbi};
                        if ($sg1 eq $en1){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$sg1\t$sg2\t$scrf1\t$scrf2\t$method\t$pubid\tIntAct-$dte\t$dte";
                                }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$sg2\t$sg1\t$scrf2\t$scrf1\t$method\t$pubid\tIntAct-$dte\t$dte";
                                }
                                print OTHER "$str_int\n";
                                push @allint_arr , $str_int ;
                        }
                }
        }
        else{
                my $kb;
                foreach $kb (keys %hsg_fbgn1){
                        my @z = split (/\t/, $kb);
                        $sg1 = $z[0];
                        $fbgn1 = $z[1];
                        $scrf1 = $hsg_fbgn1{$kb};
                        if ($sg1 eq $en1){
                                my $scrf2;
                                my $kbi;
                                foreach $kbi (keys %hsg_fbgn1){
                                        my @zi = split (/\t/, $kbi);
                                        my $sg2 = $zi[0];
                                        my $fbgn2 = $zi[1];
                                        $scrf2 = $hsg_fbgn1{$kbi};
                                        if ($sg2 eq $en2){

                                                if ($fbgn1 lt $fbgn2){
                                                        $str_int =
"$fbgn1\t$fbgn2\t$sg1\t$sg2\t$scrf1\t$scrf2\t$method\t$pubid\tIntAct-$dte\t$dte";
                                                }
                                                else{
                                                        $str_int =
"$fbgn2\t$fbgn1\t$sg2\t$sg1\t$scrf2\t$scrf1\t$method\t$pubid\tIntAct-$dte\t$dte";
                                                }
                                                print OTHER "$str_int\n";
                                                push @allint_arr , $str_int ;
                                        }
                                }
                        }
                }
```

```
		}

}

# MINT - gets all interaction data - type doesn't work - so need to make sure to only get 'association' type
print "getting MINT data\n";
$content = ();
$content =
get("http://mint.bio.uniroma2.it/mint/psicquic/webservices/current/search/query/taxidA:4932%20AND%20taxidB:493
2");
die "Couldn't get it!" unless defined $content;

#print "$content\n";
#$content =~ s/uniprotkb://g;

$content =~ s/"//g;
$content =~ s/psi-mi://g;
my $patm = qr/\|uniprotkb\:([Y|Q].*?)\(locus name\)\|/os;
my $patm0 = qr/\Auniprotkb\:([Y|Q].*?)\(locus name\)\|/os;

my @m = split (/\n/, $content);
my $rem_pat = qr/MI:0403\(colocalization\)/os; #field 12

# MINT doesn't have genetic interactions - so need to only remove colocalization
# fields 5, 6 (CG), 7 (meth), 9 pmid, date, db
foreach $ml (@m) {
		if ($ml =~ $rem_pat) {next;}
		my @in = split (/\t/, $ml);
	my $id1; my $id2;
	my $pubid;
	my $en1; my $en2;
	my $method;
		my @fb1; my @fb2;
		my $fpat = qr/uniprotkb\:([Y|Q].*?)\(locus name\)/os;
		if (($in[4] =~ $patm) || ($in[4] =~ $patm0)){
			my $i;
			my @s = split (/\|/, $in[4]);
			foreach $i (@s){
#				print "$i\n";
				if ($i =~ $fpat){
					$id1 = $1;
						last;
				}
			}
		}else {next;}
		if (($in[5] =~ $patm) ||( $in[5] =~ $patm0)){
			my $i1;
			my @s1 = split(/\|/, $in[5]);
			foreach $i1 (@s1){
#				print "$i1\n";
				if ($i1 =~ $fpat){
					$id2 =$1;
						last;
				}
			}
		}else {next;}
#		print "$id1\t$id2\n";
	if ($in[8] =~ $patpub || $in[8] =~ $patpub_unp){
		$pubid = $1;
```

```
	}else {next;}

		$method = $in[6];

	$en1 = $id1;
	$en2 = $id2;
		my $str_int;
		my $fbgn1; my $fbgn2;
		my $scrf1; my $scrf2;
		my $sg1; my $sg2;
		if ((exists $hsg_fbgn{$en1}) && (exists $hsg_fbgn{$en2})){
			@fb1 = split (/\t/, $hsg_fbgn{$en1});
			@fb2 = split (/\t/, $hsg_fbgn{$en2});
			$scrf1 = "$fb1[1]\t$fb1[2]";
			$scrf2 = "$fb2[1]\t$fb2[2]";
			$fbgn1 = $fb1[0];
			$fbgn2 = $fb2[0];
			if ($fbgn1 lt $fbgn2){
				$str_int = "$fb1[0]\t$fb2[0]\t$en1\t$en2\t$scrf1\t$scrf2\t$method\t$pubid\tMINT-
$dte\t$dte";
			}
			else{
				$str_int = "$fb2[0]\t$fb1[0]\t$en2\t$en1\t$scrf2\t$scrf1\t$method\t$pubid\tMINT-
$dte\t$dte";
			}
			print OTHER "$str_int\n";
			push @allint_arr , $str_int ;
	}
		elsif (exists $hsg_fbgn{$en1}){
			@fb1 = split (/\t/, $hsg_fbgn{$en1});
			$fbgn1 = $fb1[0];
			$sg1 = $en1;
			$scrf1 = "$fb1[1]\t$fb1[2]";
			my $kbi;
			foreach $kbi (keys %hsg_fbgn1){
				my @zi = split (/\t/, $kbi);
				$sg2 = $zi[0];
				$fbgn2 = $zi[1];
				$scrf2 = $hsg_fbgn1{$kbi};
				if ($sg2 eq $en2){
					if ($fbgn1 lt $fbgn2){
						$str_int =
"$fbgn1\t$fbgn2\t$sg1\t$sg2\t$scrf1\t$scrf2\t$method\t$pubid\tMINT-$dte\t$dte";
					}
					else{
						$str_int =
"$fbgn2\t$fbgn1\t$sg2\t$sg1\t$scrf2\t$scrf1\t$method\t$pubid\tMINT-$dte\t$dte";
					}
					print OTHER "$str_int\n";
					push @allint_arr , $str_int ;
				}
			}
		}
		elsif (exists $hsg_fbgn{$en2}){
			@fb2 = split (/\t/, $hsg_fbgn{$en2});
			$sg2 = $en2;
			$fbgn2 = $fb2[0];
			$scrf2 = "$fb2[1]\t$fb2[2]";
			my $kbi;
```

```
                foreach $kbi (keys %hsg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        $sg1 = $zi[0];
                        $fbgn1 = $zi[1];
                        $scrf1 = $hsg_fbgn1{$kbi};
                        if ($sg1 eq $en1){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$sg1\t$sg2\t$scrf1\t$scrf2\t$method\t$pubid\tMINT-$dte\t$dte";
                                }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$sg2\t$sg1\t$scrf2\t$scrf1\t$method\t$pubid\tMINT-$dte\t$dte";
                                }
                                print OTHER "$str_int\n";
                                push @allint_arr , $str_int ;
                        }
                }
        }
        else{
                my $kb;
                foreach $kb (keys %hsg_fbgn1){
                        my @z = split (/\t/, $kb);
                        $sg1 = $z[0];
                        $fbgn1 = $z[1];
                        $scrf1 = $hsg_fbgn1{$kb};
                        if ($sg1 eq $en1){
                                my $kbi;
                                foreach $kbi (keys %hsg_fbgn1){
                                        my @zi = split (/\t/, $kbi);
                                        $sg2 = $zi[0];
                                        $fbgn2 = $zi[1];
                                        $scrf2 = $hsg_fbgn1{$kbi};
                                        if ($sg2 eq $en2){

                                                if ($fbgn1 lt $fbgn2){
                                                        $str_int =
"$fbgn1\t$fbgn2\t$sg1\t$sg2\t$scrf1\t$scrf2\t$method\t$pubid\tMINT-$dte\t$dte";
                                                }
                                                else{
                                                        $str_int =
"$fbgn2\t$fbgn1\t$sg2\t$sg1\t$scrf2\t$scrf1\t$method\t$pubid\tMINT-$dte\t$dte";
                                                }
                                                print OTHER "$str_int\n";
                                                push @allint_arr , $str_int ;
                                        }
                                }
                        }
                }
        }
}

print "building table\n\n\n";
#sort so that the data is consistent with how other interaction tables are built.
@allint_arr = sort(@allint_arr);

my $item;
my %hint_s1;
my %hint_s2;
```

```
my %hint_fbs;

foreach $item (@allint_arr){
	my @a = split (/\t/, $item);
	my $fb1= $a[0];
	my $fb2 = $a[1];
		my $en1 = $a[2];
		my $en2 = $a[3];
		my $sf1 = $a[4];
		my $se1 = $a[5];
		my $sf2 = $a[6];
		my $se2 = $a[7];
	my $pubid = $a[9];
	my $method = $a[8];
	my $db = $a[10];
		my $s1 = "$en1,$se1";
		my $s2 = "$en2,$se2";
		my $fbs = "$sf1,$sf2";

	my $forw = "$fb1\t$fb2";
	if (exists $hint_db{$forw}) { push @{ $hint_db{$forw} } , $db ; }
	else { $hint_db {$forw} = [ $db ] ; }

	if (exists $hint_fbs{$forw}) { push @{ $hint_fbs{$forw} } , $fbs ; }
	else { $hint_fbs {$forw} = [ $fbs ] ; }

		if (exists $hint_s1{$forw}) { push @{ $hint_s1{$forw} } , $s1 ; }
	else { $hint_s1{$forw} = [ $s1 ] ; }

		if (exists $hint_s2{$forw}) { push @{ $hint_s2{$forw} } , $s2 ; }
	else { $hint_s2{$forw} = [ $s2 ] ; }

# pmid
	if (exists $hint_pid{$forw}){
		push @{ $hint_pid{$forw} } , $pubid ;
		my $pforw ="$forw\t$pubid";
		if (exists $hint_met{$pforw} ){
			push @{ $hint_met{$pforw} } , $method ;
		#print "$pforw\n";
		}
	}
	else {
		$hint_pid {$forw} = [ $pubid ] ;
		my $pforw ="$forw\t$pubid";
		$hint_met {$pforw} = [ $method ] ;
		#print "$pforw\n";
	}
#method
	my $pforw ="$fb1\t$fb2\t$pubid";
	if (exists $hint_met{$pforw}){ push @{ $hint_met{$pforw} } , $method ; }
	else { $hint_met {$pforw} = [ $method ] ; }
}

my $ky;
foreach $ky (keys %hint_db){
		print FINAL "$ky\t";
		my @dups_arr1 = @{ $hint_db{$ky} };
		my @no_dups1 = Remove_dups::rem_dups(@dups_arr1);
```

```
my @dups_fs = @{ $hint_fbs{$ky} };
my @no_dupsfs = Remove_dups::rem_dups(@dups_fs);

my @dups_es1 = @{ $hint_s1{$ky} };
my @no_dupes1 = Remove_dups::rem_dups(@dups_es1);

my @dups_s2 = @{ $hint_s2{$ky} };
my @no_dupes2 = Remove_dups::rem_dups(@dups_s2);

my @dups_arr2 = @{ $hint_pid{$ky} };
my @no_dups2 = Remove_dups::rem_dups(@dups_arr2);
my $pid_str = join (",", @no_dups2);
my @dups2_1 = split (/,/, $pid_str);
my @no_dups2_1 = Remove_dups::rem_dups(@dups2_1);
$pid_str = join (",", @no_dups2_1);
print FINAL "$pid_str\t";
print FINAL "$url$pid_str&dopt=DocSum\t";

my $f; #fbgn scores
my @f1; my @f2;
foreach $f (@no_dupsfs){
        my @s = split (/,/, $f);
        push @f1, $s[0];
        push @f2, $s[1];
}

my $e; # sg1 and score
my @e1; my @e2;
foreach $e (@no_dupes1){
        my @s = split (/,/, $e);
        push @e1, $s[0];
        push @e2, $s[1];
}

my $d; #sg2 and score
my @d1; my @d2;
foreach $d (@no_dupes2){
        my @s = split (/,/, $d);
        push @d1, $s[0];
        push @d2, $s[1];
}

my $str_scr = join (",", @f1);
$str_scr.= "\t";
$str_scr .= join (",", @f2);
$str_scr .= "\t";
$str_scr .= join (",", @e1);
$str_scr .= "\t";
$str_scr .= join (",", @e2);
$str_scr .= "\t";
$str_scr .= join (",", @d1);
$str_scr .= "\t";
$str_scr .= join (",", @d2);

print FINAL "$str_scr\t";

my $x;
my $met_str;
foreach $x (@no_dups2) {
```

```
                        $met_str .= $x;
#                       print "$ky, $x\n";
                        my $ky1 = "$ky\t$x";
                        my @dups_arr3 = @{ $hint_met{$ky1} };
                        my @no_dups3 = Remove_dups::rem_dups(@dups_arr3);
#                       print join("-", @no_dups3), "\n";
                        $met_str.= "(";
                        $met_str .= join ("," , @no_dups3);
                        $met_str .= "),";
            }
            $met_str = substr ($met_str, 0,length($met_str)-1);
        print FINAL "$met_str";
            print FINAL "\t", join (",", @no_dups1), "\t$dte\tpp\n";
}

close STDERR;
close OTHER;
close FINAL;
```

get_ce_interologs.pl

```
#!usr/local/bin/perl
#
#:vim syntax=perl
#
#this script uses the psicquic tool to get C.elegans data from Biogrid, intact and MINT using MIQL syntax
# read in fly FBgn, Gene Ids from output file of Inparanoid scripts and store as hash
# match IDs from these dbs through intermediate files from ncbi and ensembl to FBgns and output the interologs with
FBgns and scores etc.
# input file order entrez id file, uniprot mapping file from wormbase biomart (wbgene symbol uniprot- col order) and
finally the worm mapping output file from get_orthologs.pl

open (STDERR, ">error.out");
open (OTHER, ">ce_phy.txt");
open (FINAL, ">ce_interactions.txt");
open(INP, "<$ARGV[2]")  or die("Cannot open file '$ARGV[2]' for reading\n");
open(ENTR, "<$ARGV[0]")  or die("Cannot open file '$ARGV[0]' for reading\n");
open(UNIP, "<$ARGV[1]")  or die("Cannot open file '$ARGV[1]' for reading\n");
use strict;
use LWP::Simple;
use Thila::Remove_dups;

my %huni_wbg = ();
my %hentr_wbg = ();
my %hwbg_fbgn = ();
my %hint_db = ();
my %hint_pid = ();
my %hint_met = ();
my @allint_arr;

my $url = "http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=retrieve&db=pubmed&list_uids=";
my $cepat = qr/WormBase\:(WBGene\d+)/os;

while (<ENTR>){
    chomp $_;
    my @line = split("\t", $_);
    my $eid = $line[1];
    my $pats = $line[5];
    my $wbg;
```

```
	if ($pats =~ $cepat){
		$wbg = $1;
	}
	#this check is weird but may be necessary due to data quirks
	if (defined($wbg)){
		if ((exists $hentr_wbg{$eid}) && ( $hentr_wbg{$eid} ne $wbg)){
			$hentr_wbg{$eid} .= ", $wbg";
		}
		else { $hentr_wbg {$eid} =  $wbg;}
	}
}
close ENTR;

my $k;
foreach $k (keys %hentr_wbg){
	my $o = $hentr_wbg{$k};
	if ($o =~ /,/){ delete $hentr_wbg{$k}; }
}

while (<UNIP>){
	chomp $_;
	if ($_ =~ /WBGene\d+/ ){
		my @line = split(/\t/, $_);
		my $wbg = $line[0];
		my $uni = $line[2];
		if (defined($uni)){
			if ((exists $huni_wbg{$uni}) && ( $huni_wbg{$uni} ne $wbg)){
			$huni_wbg{$uni} .= ", $wbg";
		}
		else { $huni_wbg {$uni} =  $wbg;}

		}
	}else {next;}
}
close UNIP;

my $i;
foreach $i (keys %huni_wbg){
	my $o = $huni_wbg{$i};
	if ($o =~ /,/){ delete $huni_wbg{$i};  }
}

while (<INP>){
	chomp $_;
	my @line = split (/\t/, $_);
	my $fb = $line[2];
	my $fp = $line[3];
	my $scr1 = $line[4];
	my $en = $line[5];
	my $enp = $line[6];
	my $scr2 = $line[7];
	my $h = "$en";
		my $scr = "$fb\t$scr1\t$scr2";
	if ((exists $hwbg_fbgn{$h}) && ( $hwbg_fbgn{$h} ne $scr)){
		$hwbg_fbgn{$h} .= ",$scr";
	}
	else { $hwbg_fbgn {$h} =  $scr;}
}
my $a;
```

```
my %hwbg_fbgn1;
#foreach $a1 ( keys %hensg_fbgn) {
#	print OUT3 "$a1\t", join ("\t", $hensg_fbgn{$a1} ), "\n";
#}

foreach $a (keys %hwbg_fbgn){
	my $o = $hwbg_fbgn{$a};
	if ($o =~ /,/){
			my @e = split (/,/, $o);
			my $i;
			foreach $i (@e){
				my @ei = split (/\t/, $i);
				my $r = "$a\t$ei[0]";
				$hwbg_fbgn1{$r} = "$ei[1]\t$ei[2]";
			}
			delete $hwbg_fbgn{$a};
		}
}

close INP;

my $curr_day = (localtime)[3];
my @mon = ("JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", "DEC");
my $curr_mon = (localtime)[4];
my $curr_year = (localtime)[5]+1900;
my $dte= "$curr_day-";
$dte .=$mon[$curr_mon];
$dte .= "-";
$dte .= substr $curr_year, 2;
print "$dte -made hashes. Getting BioGrid data\n";

my $bl;
my $il;
my $ml;
my $content = ();
# biogrid get all physical interactions (MI OBO term 'association')
# AND both belonging to worm

$content = get("http://tyerslab.bio.ed.ac.uk:8080/psicquic-
ws/webservices/current/search/query/type:%22*association%22%20AND%20taxidA:6239%20AND%20taxidB:6239"
);
die "Couldn't get it!" unless defined $content;
#print "$content\n";
my $patb = qr/entrez gene\/locuslink:(\d+)/os;
my $patpub = qr/pubmed:(\d+)/os;
my $patpub_unp = qr/pubmed:(\w+.*?)/os;

$content =~ s/"//g;
$content =~ s/psi-mi://g;

my @b = split (/\n/, $content);
#fields 3, 4 sym, 5, 6 CG, 7 meth, 9 pmid, add date download, db
foreach $bl (@b){
		my @in = split (/\t/, $bl);
		my $id1; my $id2;
		my $pubid;
		my $en1; my $en2;
		my $method;
		my @fb1; my @fb2;
```

```
        if ($in[0] =~ $patb) {
            $id1 = $1;
        }
        else {next;}
        if ($in[1] =~ $patb){
            $id2= $1;
        }
        else {next;}
        if ($in[8] =~ $patpub || $in[8] =~ $patpub_unp){
                $pubid = $1;
        } else {next;}
        $method = $in[6];

        if ((exists $hentr_wbg{$id1}) && (exists $hentr_wbg{$id2})){
                $en1 = $hentr_wbg{$id1};
                $en2 = $hentr_wbg{$id2};
    } else {next;}
        my $str_int;
        if ((exists $hwbg_fbgn{$en1}) && (exists $hwbg_fbgn{$en2})){
                @fb1 = split (/\t/, $hwbg_fbgn{$en1});
                @fb2 = split (/\t/, $hwbg_fbgn{$en2});
                my $scrf1 = "$fb1[1]\t$fb1[2]";
                my $scrf2 = "$fb2[1]\t$fb2[2]";
                my $fbgn1 = $fb1[0];
                my $fbgn2 = $fb2[0];
                if ($fbgn1 lt $fbgn2){
                        $str_int = "$fb1[0]\t$fb2[0]\t$en1\t$en2\t$scrf1\t$scrf2\t$method\t$pubid\tBioGRID-
$dte\t$dte";
                }
                else{
                        $str_int = "$fb2[0]\t$fb1[0]\t$en2\t$en1\t$scrf2\t$scrf1\t$method\t$pubid\tBioGRID-
$dte\t$dte";
                }
                print OTHER "$str_int\n";
                push @allint_arr , $str_int ;
    }
        elsif (exists $hwbg_fbgn{$en1}){
                @fb1 = split (/\t/, $hwbg_fbgn{$en1});
                my $fbgn1 = $fb1[0];
                my $scrf1 = "$fb1[1]\t$fb1[2]";
                my $wbg1 = $en1;
                my $kbi;
                foreach $kbi (keys %hwbg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        my $wbg2 = $zi[0];
                        my $fbgn2 = $zi[1];
                        my $scrf2 = $hwbg_fbgn1{$kbi};
                        if ($wbg2 eq $en2){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$wbg1\t$wbg2\t$scrf1\t$scrf2\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$wbg2\t$wbg1\t$scrf2\t$scrf1\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                }
                                print OTHER "$str_int\n";
                                push @allint_arr , $str_int ;
```

```
                        }
                }
        }
        elsif (exists $hwbg_fbgn{$en2}){
                @fb2 = split (/\t/, $hwbg_fbgn{$en2});
                my $wbg2 = $en2;
                my $fbgn2 = $fb2[0];
                my $scrf2 = "$fb2[1]\t$fb2[2]";
                my $kbi;
                foreach $kbi (keys %hwbg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        my $wbg1 = $zi[0];
                        my $fbgn1 = $zi[1];
                        my $scrf1 = $hwbg_fbgn1{$kbi};
                        if ($wbg1 eq $en1){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$wbg1\t$wbg2\t$scrf1\t$scrf2\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$wbg2\t$wbg1\t$scrf2\t$scrf1\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                }
                                print OTHER "$str_int\n";
                                push @allint_arr , $str_int ;
                        }
                }
        }
        else {
                my $kb;
                foreach $kb (keys %hwbg_fbgn1){
                        my @z = split (/\t/, $kb);
                        my $wbg1 = $z[0];
                        my $fbgn1 = $z[1];
                        my $scrf1 = $hwbg_fbgn1{$kb};
                        if ($wbg1 eq $en1){
                                my $scrf2;
                                my $kbi;
                                foreach $kbi (keys %hwbg_fbgn1){
                                        my @zi = split (/\t/, $kbi);
                                        my $wbg2 = $zi[0];
                                        my $fbgn2 = $zi[1];
                                        $scrf2 = $hwbg_fbgn1{$kbi};
                                        if ($wbg2 eq $en2){
                                                if ($fbgn1 lt $fbgn2){
                                                        $str_int =
"$fbgn1\t$fbgn2\t$wbg1\t$wbg2\t$scrf1\t$scrf2\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                                }
                                                else{
                                                        $str_int =
"$fbgn2\t$fbgn1\t$wbg2\t$wbg1\t$scrf2\t$scrf1\t$method\t$pubid\tBioGRID-$dte\t$dte";
                                                }
                                                print OTHER "$str_int\n";
                                                push @allint_arr , $str_int ;
                                        }
                                }
                        }
                }
        }
```

```
}
#intact data
print "getting intact data\n";
$content = ();
$content =
get("http://www.ebi.ac.uk/Tools/webservices/psicquic/intact/webservices/current/search/query/type:%22*association%
22%20AND%20taxidA:6239%20AND%20taxidB:6239");
die "Couldn't get it!" unless defined $content;
#$content =~ s/uniprotkb://g;
$content =~ s/"//g;
$content =~ s/psi-mi://g;

#print "$content\n";
#my $pati0 = qr/wormbase\:(WBGene\d+)/os;
my $pati0 = qr/uniprotkb\:(.*?)\|intact/os;
my @i = split (/\n/, $content);
foreach $il (@i) {
        my @in = split (/\t/, $il);
    my $en1; my $en2;
    my $pubid;
    my $method;
        my @fb1; my @fb2;
        my $id1; my $id2;
#     if ($in[19] =~ $pati0) { #changed in jan2011
        #       $en1 = $1;
        #}else {next;}

        #if ($in[20] =~ $pati0){
        #   $en2= $1;
        #}else {next;}
        if ($in[0] =~ $pati0) {
      $id1 = $1;
        }else {next;}

        if ($in[1] =~ $pati0){
            $id2= $1;
        }else {next;}

    if ($in[8] =~ $patpub || $in[8] =~ $patpub_unp){
        $pubid = $1;
    } else {next;}

        $method = $in[6];
        if ((exists $huni_wbg{$id1}) && (exists $huni_wbg{$id2})){
        $en1 = $huni_wbg{$id1};
        $en2 = $huni_wbg{$id2};
    } else {next;}
        my $str_int;
        my $fbgn1; my $fbgn2;
        my $scrf1; my $scrf2;
        my $wbg1; my $wbg2;
        if ((exists $hwbg_fbgn{$en1}) && (exists $hwbg_fbgn{$en2})){
                @fb1 = split (/\t/, $hwbg_fbgn{$en1});
                @fb2 = split (/\t/, $hwbg_fbgn{$en2});
                $fbgn1 = $fb1[0];
                $fbgn2 = $fb2[0];
                $scrf1 = "$fb1[1]\t$fb1[2]";
                $scrf2 = "$fb2[1]\t$fb2[2]";
                if ($fbgn1 lt $fbgn2){
```

```
                        $str_int = "$fb1[0]\t$fb2[0]\t$en1\t$en2\t$scrf1\t$scrf2\t$method\t$pubid\tIntAct-
$dte\t$dte";
                }
                else{
                        $str_int = "$fb2[0]\t$fb1[0]\t$en2\t$en1\t$scrf2\t$scrf1\t$method\t$pubid\tIntAct-
$dte\t$dte";
                }
                print OTHER "$str_int\n";
                push @allint_arr , $str_int ;
        }
        elsif (exists $hwbg_fbgn{$en1}){
                @fb1 = split (/\t/, $hwbg_fbgn{$en1});
                $fbgn1 = $fb1[0];
                $wbg1 = $en1;
                $scrf1 = "$fb1[1]\t$fb1[2]";
                my $kbi;
                foreach $kbi (keys %hwbg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        $wbg2 = $zi[0];
                        $fbgn2 = $zi[1];
                        $scrf2 = $hwbg_fbgn1{$kbi};
                        if ($wbg2 eq $en2){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$wbg1\t$wbg2\t$scrf1\t$scrf2\t$method\t$pubid\tIntAct-$dte\t$dte";
                                }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$wbg2\t$wbg1\t$scrf2\t$scrf1\t$method\t$pubid\tIntAct-$dte\t$dte";
                                }
                                print OTHER "$str_int\n";
                                push @allint_arr , $str_int ;
                        }
                }
        }
        elsif (exists $hwbg_fbgn{$en2}){
                @fb2 = split (/\t/, $hwbg_fbgn{$en2});
                $wbg2 = $en2;
                $fbgn2 = $fb2[0];
                $scrf2 = "$fb2[1]\t$fb2[2]";
                my $kbi;
                foreach $kbi (keys %hwbg_fbgn1){
                        my @zi = split (/\t/, $kbi);
                        $wbg1 = $zi[0];
                        $fbgn1 = $zi[1];
                        $scrf1 = $hwbg_fbgn1{$kbi};
                        if ($wbg1 eq $en1){
                                if ($fbgn1 lt $fbgn2){
                                        $str_int =
"$fbgn1\t$fbgn2\t$wbg1\t$wbg2\t$scrf1\t$scrf2\t$method\t$pubid\tIntAct-$dte\t$dte";
                                }
                                else{
                                        $str_int =
"$fbgn2\t$fbgn1\t$wbg2\t$wbg1\t$scrf2\t$scrf1\t$method\t$pubid\tIntAct-$dte\t$dte";
                                }
                                print OTHER "$str_int\n";
                                push @allint_arr , $str_int ;
                        }
                }
```

```
		}
		else{
			my $kb;
			foreach $kb (keys %hwbg_fbgn1){
				my @z = split (/\t/, $kb);
				$wbg1 = $z[0];
				$fbgn1 = $z[1];
				$scrf1 = $hwbg_fbgn1{$kb};
				if ($wbg1 eq $en1){
					my $scrf2;
					my $kbi;
					foreach $kbi (keys %hwbg_fbgn1){
						my @zi = split (/\t/, $kbi);
						my $wbg2 = $zi[0];
						my $fbgn2 = $zi[1];
						$scrf2 = $hwbg_fbgn1{$kbi};
						if ($wbg2 eq $en2){

							if ($fbgn1 lt $fbgn2){
								$str_int =
"$fbgn1\t$fbgn2\t$wbg1\t$wbg2\t$scrf1\t$scrf2\t$method\t$pubid\tIntAct-$dte\t$dte";
							}
							else{
								$str_int =
"$fbgn2\t$fbgn1\t$wbg2\t$wbg1\t$scrf2\t$scrf1\t$method\t$pubid\tIntAct-$dte\t$dte";
							}
							print OTHER "$str_int\n";
							push @allint_arr , $str_int ;
						}
					}
				}
			}
		}

}

# MINT - gets all interaction data - type doesn't work - so need to make sure to only get 'association' type
print "getting MINT data\n";
$content = ();
$content =
get("http://mint.bio.uniroma2.it/mint/psicquic/webservices/current/search/query/taxidA:6239%20AND%20taxidB:623
9");
die "Couldn't get it!" unless defined $content;

#print "$content\n";
$content =~ s/uniprotkb://g;

$content =~ s/"//g;
$content =~ s/psi-mi://g;

my @m = split (/\n/, $content);
my $rem_pat = qr/MI:0403\(colocalization\)/os; #field 12

# MINT doesn't have genetic interactions - so need to only remove colocalization
# fields 5, 6 (CG), 7 (meth), 9 pmid, date, db
foreach $ml (@m) {
		if ($ml =~ $rem_pat) {next;}
		my @in = split (/\t/, $ml);
	my $id1; my $id2;
```

```
	my $pubid;
	my $en1; my $en2;
	my $method;
		my @fb1; my @fb2;

	$id1 = $in[0];

	$id2 =$in[1];
#		print "$id1\t$id2\n";
	if ($in[8] =~ $patpub || $in[8] =~ $patpub_unp){
		$pubid = $1;
	}else {next;}

		$method = $in[6];

		if ((exists $huni_wbg{$id1}) && (exists $huni_wbg{$id2})){
		$en1 = $huni_wbg{$id1};
		$en2 = $huni_wbg{$id2};
	} else {next;}
		my $str_int;
		my $fbgn1; my $fbgn2;
		my $scrf1; my $scrf2;
		my $wbg1; my $wbg2;
		if ((exists $hwbg_fbgn{$en1}) && (exists $hwbg_fbgn{$en2})){
			@fb1 = split (/\t/, $hwbg_fbgn{$en1});
			@fb2 = split (/\t/, $hwbg_fbgn{$en2});
			$scrf1 = "$fb1[1]\t$fb1[2]";
			$scrf2 = "$fb2[1]\t$fb2[2]";
			$fbgn1 = $fb1[0];
			$fbgn2 = $fb2[0];
			if ($fbgn1 lt $fbgn2){
				$str_int = "$fb1[0]\t$fb2[0]\t$en1\t$en2\t$scrf1\t$scrf2\t$method\t$pubid\tMINT-
$dte\t$dte";
			}
			else{
				$str_int = "$fb2[0]\t$fb1[0]\t$en2\t$en1\t$scrf2\t$scrf1\t$method\t$pubid\tMINT-
$dte\t$dte";
			}
			print OTHER "$str_int\n";
			push @allint_arr , $str_int ;
	}
		elsif (exists $hwbg_fbgn{$en1}){
			@fb1 = split (/\t/, $hwbg_fbgn{$en1});
			$fbgn1 = $fb1[0];
			$wbg1 = $en1;
			$scrf1 = "$fb1[1]\t$fb1[2]";
			my $kbi;
			foreach $kbi (keys %hwbg_fbgn1){
				my @zi = split (/\t/, $kbi);
				$wbg2 = $zi[0];
				$fbgn2 = $zi[1];
				$scrf2 = $hwbg_fbgn1{$kbi};
				if ($wbg2 eq $en2){
					if ($fbgn1 lt $fbgn2){
						$str_int =
"$fbgn1\t$fbgn2\t$wbg1\t$wbg2\t$scrf1\t$scrf2\t$method\t$pubid\tMINT-$dte\t$dte";
					}
					else{
```

```
                                                $str_int =
"$fbgn2\t$fbgn1\t$wbg2\t$wbg1\t$scrf2\t$scrf1\t$method\t$pubid\tMINT-$dte\t$dte";
                                        }
                                        print OTHER "$str_int\n";
                                        push @allint_arr , $str_int ;
                                }
                        }
                }
                elsif (exists $hwbg_fbgn{$en2}){
                        @fb2 = split (/\t/, $hwbg_fbgn{$en2});
                        $wbg2 = $en2;
                        $fbgn2 = $fb2[0];
                        $scrf2 = "$fb2[1]\t$fb2[2]";
                        my $kbi;
                        foreach $kbi (keys %hwbg_fbgn1){
                                my @zi = split (/\t/, $kbi);
                                $wbg1 = $zi[0];
                                $fbgn1 = $zi[1];
                                $scrf1 = $hwbg_fbgn1{$kbi};
                                if ($wbg1 eq $en1){
                                        if ($fbgn1 lt $fbgn2){
                                                $str_int =
"$fbgn1\t$fbgn2\t$wbg1\t$wbg2\t$scrf1\t$scrf2\t$method\t$pubid\tMINT-$dte\t$dte";
                                        }
                                        else{
                                                $str_int =
"$fbgn2\t$fbgn1\t$wbg2\t$wbg1\t$scrf2\t$scrf1\t$method\t$pubid\tMINT-$dte\t$dte";
                                        }
                                        print OTHER "$str_int\n";
                                        push @allint_arr , $str_int ;
                                }
                        }
                }
                else{
                        my $kb;
                        foreach $kb (keys %hwbg_fbgn1){
                                my @z = split (/\t/, $kb);
                                $wbg1 = $z[0];
                                $fbgn1 = $z[1];
                                $scrf1 = $hwbg_fbgn1{$kb};
                                if ($wbg1 eq $en1){
                                        my $kbi;
                                        foreach $kbi (keys %hwbg_fbgn1){
                                                my @zi = split (/\t/, $kbi);
                                                $wbg2 = $zi[0];
                                                $fbgn2 = $zi[1];
                                                $scrf2 = $hwbg_fbgn1{$kbi};
                                                if ($wbg2 eq $en2){

                                                        if ($fbgn1 lt $fbgn2){
                                                                $str_int =
"$fbgn1\t$fbgn2\t$wbg1\t$wbg2\t$scrf1\t$scrf2\t$method\t$pubid\tMINT-$dte\t$dte";
                                                        }
                                                        else{
                                                                $str_int =
"$fbgn2\t$fbgn1\t$wbg2\t$wbg1\t$scrf2\t$scrf1\t$method\t$pubid\tMINT-$dte\t$dte";
                                                        }
                                                        print OTHER "$str_int\n";
                                                        push @allint_arr , $str_int ;
```

```
                                                    }
                                                }
                                            }
                                        }
                                }
}

print "building table\n\n\n";
#sort so that the data is consistent with how other interaction tables are built.
@allint_arr = sort(@allint_arr);

my $item;
my %hint_ens1;
my %hint_ens2;
my %hint_fbs;

foreach $item (@allint_arr){
        my @a = split (/\t/, $item);
        my $fb1= $a[0];
        my $fb2 = $a[1];
                my $en1 = $a[2];
                my $en2 = $a[3];
                my $sf1 = $a[4];
                my $se1 = $a[5];
                my $sf2 = $a[6];
                my $se2 = $a[7];
        my $pubid = $a[9];
        my $method = $a[8];
        my $db = $a[10];
                my $ens1 = "$en1,$se1";
                my $ens2 = "$en2,$se2";
                my $fbs = "$sf1,$sf2";

        my $forw = "$fb1\t$fb2";
        if (exists $hint_db{$forw}) { push @{ $hint_db{$forw} } , $db ; }
        else { $hint_db {$forw} = [ $db ] ; }

        if (exists $hint_fbs{$forw}) { push @{ $hint_fbs{$forw} } , $fbs ; }
        else { $hint_fbs {$forw} = [ $fbs ] ; }

                if (exists $hint_ens1{$forw}) { push @{ $hint_ens1{$forw} } , $ens1 ; }
        else { $hint_ens1{$forw} = [ $ens1 ] ; }

                if (exists $hint_ens2{$forw}) { push @{ $hint_ens2{$forw} } , $ens2 ; }
        else { $hint_ens2{$forw} = [ $ens2 ] ; }

# pmid
        if (exists $hint_pid{$forw}){
                push @{ $hint_pid{$forw} } , $pubid ;
                my $pforw ="$forw\t$pubid";
                if (exists $hint_met{$pforw} ){
                                push @{ $hint_met{$pforw} } , $method ;
                #print "$pforw\n";
                }
        }
        else {
                $hint_pid {$forw} = [ $pubid ] ;
                my $pforw ="$forw\t$pubid";
                $hint_met {$pforw} = [ $method ] ;
```

```
            #print "$pforw\n";
        }
#method
        my $pforw ="$fb1\t$fb2\t$pubid";
        if (exists $hint_met{$pforw}){ push @{ $hint_met{$pforw} } , $method ; }
        else { $hint_met {$pforw} = [ $method ] ; }
}

my $ky;
foreach $ky (keys %hint_db){
            print FINAL "$ky\t";
            my @dups_arr1 = @{ $hint_db{$ky} };
            my @no_dups1 = Remove_dups::rem_dups(@dups_arr1);

            my @dups_fs = @{ $hint_fbs{$ky} };
            my @no_dupsfs = Remove_dups::rem_dups(@dups_fs);

            my @dups_es1 = @{ $hint_ens1{$ky} };
            my @no_dupes1 = Remove_dups::rem_dups(@dups_es1);

            my @dups_ens2 = @{ $hint_ens2{$ky} };
            my @no_dupes2 = Remove_dups::rem_dups(@dups_ens2);

            my @dups_arr2 = @{ $hint_pid{$ky} };
            my @no_dups2 = Remove_dups::rem_dups(@dups_arr2);
            my $pid_str = join (",", @no_dups2);
            my @dups2_1 = split (/,/, $pid_str);
            my @no_dups2_1 = Remove_dups::rem_dups(@dups2_1);
            $pid_str = join (",", @no_dups2_1);
            print FINAL "$pid_str\t";
            print FINAL "$url$pid_str&dopt=DocSum\t";

            my $f; #fbgn scores
            my @f1; my @f2;
            foreach $f (@no_dupsfs){
                        my @s = split (/,/, $f);
                        push @f1, $s[0];
                        push @f2, $s[1];
            }

            my $e; # wbg1 and score
            my @e1; my @e2;
            foreach $e (@no_dupes1){
                        my @s = split (/,/, $e);
                        push @e1, $s[0];
                        push @e2, $s[1];
            }

            my $d; #wbg2 and score
            my @d1; my @d2;
            foreach $d (@no_dupes2){
                        my @s = split (/,/, $d);
                        push @d1, $s[0];
                        push @d2, $s[1];
            }

            my $str_scr = join (",", @f1);
            $str_scr.= "\t";
            $str_scr .= join (",", @f2);
```

```
		$str_scr .= "\t";
		$str_scr .= join (",", @e1);
		$str_scr .= "\t";
		$str_scr .= join (",", @e2);
		$str_scr .= "\t";
		$str_scr .= join (",", @d1);
		$str_scr .= "\t";
		$str_scr .= join (",", @d2);

		print FINAL "$str_scr\t";

		my $x;
		my $met_str;
		foreach $x (@no_dups2) {
			$met_str .= $x;
#			print "$ky, $x\n";
			my $ky1 = "$ky\t$x";
			my @dups_arr3 = @{ $hint_met{$ky1} };
			my @no_dups3 = Remove_dups::rem_dups(@dups_arr3);
#			print join("-", @no_dups3), "\n";
			$met_str.= "(";
			$met_str .= join ("," , @no_dups3);
			$met_str .= "),";
		}
		$met_str = substr ($met_str, 0,length($met_str)-1);
	print FINAL "$met_str";
		print FINAL "\t", join (",", @no_dups1), "\t$dte\tpp\n";
}

close STDERR;
close OTHER;
close FINAL;
```

APPENDIX C

Step 1. Execute delt_unmap;
Step 2. Execute upd_fbgn_mapping;
Step 3. Execute ins_corr;
Step 4. Execute upd_syms;
*Note clean_tmp is a required subroutine which is called from within other procedures.

```
create or replace PROCEDURE delt_unmap
IS
BEGIN
/* 1. copy all interaction fbgns into tmp table with 4 cols fbgn1_new, fbgn1_old, fbgn2_new, fbgn2_old and run proc cleantmp*/
/* 2. This proc is for delete after finding not mappable interactions.
a)          link tbl of choice on not_mappable and copy to tbl_man_curate for yth and then run a delete
b)          link tbl of choice on tmp_holder (old fbgns) and copy to tbl_man_curate for yth and then run a delete*/
/*human interologs*/
DELETE FROM tmp_holder;
INSERT INTO tmp_holder (fbgn1_old, fbgn1_new, fbgn2_old, fbgn2_new) (select fly_gene1, fly_gene1 as fbgn1new, fly_gene2, fly_gene2 as fbgn2new from human_interologs);
clean_tmp; /*ids duplicates*/
DELETE from human_interologs where (Fly_GENE1, Fly_GENE2) in (SELECT fbgn1_old, fbgn2_old FROM tmp_holder);
DELETE from human_interologs where ((Fly_GENE1) in (SELECT secfbgn FROM not_mappable) OR (Fly_GENE2) in (SELECT secfbgn FROM not_mappable));
/*other physical*/
DELETE FROM tmp_holder;
INSERT INTO tmp_holder (fbgn1_old, fbgn1_new, fbgn2_old, fbgn2_new) (select fly_gene1, fly_gene1 as fbgn1new, fly_gene2, fly_gene2 as fbgn2new from fly_other_physical);
clean_tmp; /*ids duplicates*/
DELETE from fly_other_physical where (Fly_GENE1, Fly_GENE2) in (SELECT fbgn1_old, fbgn2_old FROM tmp_holder);
DELETE from fly_other_physical where ((Fly_GENE1) in (SELECT secfbgn FROM not_mappable) OR (Fly_GENE2) in (SELECT secfbgn FROM not_mappable));
/* yeast interologs */
DELETE FROM tmp_holder;
INSERT INTO tmp_holder (fbgn1_old, fbgn1_new, fbgn2_old, fbgn2_new) (select fly_gene1, fly_gene1 as fbgn1new, fly_gene2, fly_gene2 as fbgn2new from yeast_interologs);
clean_tmp;
DELETE from yeast_interologs where (Fly_GENE1, Fly_GENE2) in (SELECT fbgn1_old, fbgn2_old FROM tmp_holder);
DELETE from yeast_interologs where ((Fly_GENE1) in (SELECT secfbgn FROM not_mappable) OR (Fly_GENE2) in (SELECT secfbgn FROM not_mappable));
/*worm interologs */
DELETE FROM tmp_holder;
INSERT INTO tmp_holder (fbgn1_old, fbgn1_new, fbgn2_old, fbgn2_new) (select fly_gene1, fly_gene1 as fbgn1new, fly_gene2, fly_gene2 as fbgn2new from worm_interologs);
clean_tmp;
DELETE from worm_interologs where (Fly_GENE1, Fly_GENE2) in (SELECT fbgn1_old, fbgn2_old FROM tmp_holder);
DELETE from worm_interologs where ((Fly_GENE1) in (SELECT secfbgn FROM not_mappable) OR (Fly_GENE2) in (SELECT secfbgn FROM not_mappable));
/*TF gene */
DELETE FROM tmp_holder;
INSERT INTO tmp_holder (fbgn1_old, fbgn1_new, fbgn2_old, fbgn2_new) (select fly_tf_gene, fly_tf_gene as fbgn1new, fly_target_gene, fly_target_gene as fbgn2new from tf_gene);
clean_tmp;
DELETE from tf_gene where (Fly_tf_GENE, Fly_target_GENE) in (SELECT fbgn1_old, fbgn2_old FROM tmp_holder);
DELETE from tf_gene where ((Fly_tf_GENE) in (SELECT secfbgn FROM not_mappable) OR (Fly_target_GENE) in (SELECT secfbgn FROM not_mappable));
/*RNA gene */
DELETE FROM tmp_holder;
```

```
INSERT INTO tmp_holder (fbgn1_old, fbgn1_new, fbgn2_old, fbgn2_new) (select fly_rna_gene, fly_rna_gene as fbgn1new, fly_target_gene, fly_target_gene as fbgn2new from rna_gene);
clean_tmp;
DELETE from rna_gene where (Fly_rna_GENE, Fly_target_GENE) in (SELECT fbgn1_old, fbgn2_old FROM tmp_holder);
DELETE from rna_gene where ((Fly_rna_GENE) in (SELECT secfbgn FROM not_mappable) OR (Fly_target_GENE) in (SELECT secfbgn FROM not_mappable));
/*friedmanperrimon coap */
DELETE FROM tmp_holder;
INSERT INTO tmp_holder (fbgn1_old, fbgn1_new, fbgn2_old, fbgn2_new) (select fbgn_bait, fbgn_bait as fbgn1new, fbgn_interactor, fbgn_interactor as fbgn2new from friedmanperrimon_coap);
clean_tmp;
DELETE from friedmanperrimon_coap where (Fbgn_bait, Fbgn_interactor) in (SELECT fbgn1_old, fbgn2_old FROM tmp_holder);
DELETE from friedmanperrimon_coap where ((Fbgn_bait) in (SELECT secfbgn FROM not_mappable) OR (Fbgn_interactor) in (SELECT secfbgn FROM not_mappable));
/*dpim coapcomplex */
DELETE FROM tmp_holder;
INSERT INTO tmp_holder (fbgn1_old, fbgn1_new, fbgn2_old, fbgn2_new) (select fbgn_bait, fbgn_bait as fbgn1new, fbgn_interactor, fbgn_interactor as fbgn2new from dpim_coapcomplex);
clean_tmp;
DELETE from dpim_coapcomplex where (Fbgn_bait, Fbgn_interactor) in (SELECT fbgn1_old, fbgn2_old FROM tmp_holder);
DELETE from dpim_coapcomplex where ((Fbgn_bait) in (SELECT secfbgn FROM not_mappable) OR (Fbgn_interactor) in (SELECT secfbgn FROM not_mappable));
/*correlation */
DELETE FROM correlation WHERE RWEIGHTED is null AND CONFIDENCE is null;
DELETE FROM tmp_holder;
INSERT INTO tmp_holder (fbgn1_old, fbgn1_new, fbgn2_old, fbgn2_new) (select fly_gene1, fly_gene1 as fbgn1new, fly_gene2, fly_gene2 as fbgn2new from correlation);
clean_tmp;
DELETE from correlation where (Fly_GENE1, Fly_GENE2) in (SELECT fbgn1_old, fbgn2_old FROM tmp_holder);
DELETE from correlation where ((Fly_GENE1) in (SELECT secfbgn FROM not_mappable) OR (Fly_GENE2) in (SELECT secfbgn FROM not_mappable));
/* curagen yth*/
DELETE FROM tmp_holder;
INSERT INTO tmp_holder (fbgn1_old, fbgn1_new, fbgn2_old, fbgn2_new) (select FBGN_GENE1_BD, FBGN_GENE1_BD as fbgn1new, FBGN_GENE2_AD, FBGN_GENE2_AD as fbgn2new from curagen_yth);
clean_tmp;
INSERT INTO tbl_man_curate (fbgn1, fbgn2, therest, tbl_name, date_rem) (SELECT FBGN_GENE1_BD, FBGN_GENE2_AD, GENE1_INTERACTIONS_AS_BD || ', '|| GENE1_INTERACTIONS_TOTAL || ', '|| GENE2_INTERACTIONS_AS_AD || ', '|| GENE2_INTERACTIONS_TOTAL || ', '|| SCREEN || ', '|| REFERENCE || ', '|| DATE1 || ', '|| CDNA || ', '|| COLLECTION || ', '|| HEXPERT || ', '|| YEXPERT || ', '|| CEXPERT || ', '|| CURAGEN_CONFIDENCE || ', '|| ISTS_RFCS || ', '|| DATA_VERSION || ', '|| DATE_LAST_UPDATED as therest, upper('curagen yth'), SYSDATE FROM curagen_yth, tmp_holder WHERE FBGN_GENE1_BD = fbgn1_old AND FBGN_GENE2_AD = fbgn2_old);
DELETE from curagen_yth where (FBGN_GENE1_BD, FBGN_GENE2_AD) in (SELECT fbgn1_old, fbgn2_old FROM tmp_holder);
INSERT INTO tbl_man_curate (fbgn1, fbgn2, therest, tbl_name, date_rem) (SELECT FBGN_GENE1_BD, FBGN_GENE2_AD, GENE1_INTERACTIONS_AS_BD || ', '|| GENE1_INTERACTIONS_TOTAL || ', '|| GENE2_INTERACTIONS_AS_AD || ', '|| GENE2_INTERACTIONS_TOTAL || ', '|| SCREEN || ', '|| REFERENCE || ', '|| DATE1 || ', '|| CDNA || ', '|| COLLECTION || ', '|| HEXPERT || ', '|| YEXPERT || ', '|| CEXPERT || ', '|| CURAGEN_CONFIDENCE || ', '|| ISTS_RFCS || ', '|| DATA_VERSION || ', '|| DATE_LAST_UPDATED as therest, upper('curagen yth'), SYSDATE FROM curagen_yth, not_mappable WHERE FBGN_GENE1_BD = secfbgn or FBGN_GENE2_AD = secfbgn);
DELETE from curagen_yth where ((FBGN_GENE1_BD) in (SELECT secfbgn FROM not_mappable) OR (FBGN_GENE2_AD) in (SELECT secfbgn FROM not_mappable));
/* finley yth */
DELETE FROM tmp_holder;
INSERT INTO tmp_holder (fbgn1_old, fbgn1_new, fbgn2_old, fbgn2_new) (select FBGN_GENE1_BD, FBGN_GENE1_BD as fbgn1new, FBGN_GENE2_AD, FBGN_GENE2_AD as fbgn2new from finley_yth);
clean_tmp;
```

```
INSERT INTO tbl_man_curate (fbgn1, fbgn2, therest, tbl_name, date_rem) (SELECT FBGN_GENE1_BD, FBGN_GENE2_AD, GENE1_INTERACTIONS_AS_BD || ', '|| GENE1_INTERACTIONS_TOTAL || ', '|| GENE2_INTERACTIONS_AS_AD || ', '|| GENE2_INTERACTIONS_TOTAL || ', '|| SCREEN || ', '|| REFERENCE || ', '|| DATE1 || ', '|| C_LEU || ', '|| C_LACZ || ', '|| C_SUM || ', '|| MATRIX || ', '|| IST || ', '|| MATRIX_DETECTIONS || ', '|| ISTS_RFCS || ', '|| DATA_VERSION || ', '|| DATE_LAST_UPDATED as therest, upper('finley yth'), SYSDATE FROM finley_yth, tmp_holder WHERE FBGN_GENE1_BD = fbgn1_old AND FBGN_GENE2_AD = fbgn2_old);
DELETE from finley_yth where (FBGN_GENE1_BD, FBGN_GENE2_AD) in (SELECT fbgn1_old, fbgn2_old FROM tmp_holder);
INSERT INTO tbl_man_curate (fbgn1, fbgn2, therest, tbl_name, date_rem) (SELECT FBGN_GENE1_BD, FBGN_GENE2_AD, GENE1_INTERACTIONS_AS_BD || ', '|| GENE1_INTERACTIONS_TOTAL || ', '|| GENE2_INTERACTIONS_AS_AD || ', '|| GENE2_INTERACTIONS_TOTAL || ', '|| SCREEN || ', '|| REFERENCE || ', '|| DATE1 || ', '|| C_LEU || ', '|| C_LACZ || ', '|| C_SUM || ', '|| MATRIX || ', '|| IST || ', '|| MATRIX_DETECTIONS || ', '|| ISTS_RFCS || ', '|| DATA_VERSION || ', '|| DATE_LAST_UPDATED as therest, upper('finley yth'), SYSDATE FROM finley_yth, not_mappable WHERE FBGN_GENE1_BD = secfbgn or FBGN_GENE2_AD = secfbgn);
DELETE from finley_yth where ((FBGN_GENE1_BD) in (SELECT secfbgn FROM not_mappable) OR (FBGN_GENE2_AD) in (SELECT secfbgn FROM not_mappable));
/* hybrigenics yth */
DELETE FROM tmp_holder;
INSERT INTO tmp_holder (fbgn1_old, fbgn1_new, fbgn2_old, fbgn2_new) (select FBGN_GENE1_BD, FBGN_GENE1_BD as fbgn1new, FBGN_GENE2_AD, FBGN_GENE2_AD as fbgn2new from hybrigenics_yth);
clean_tmp;
INSERT INTO tbl_man_curate (fbgn1, fbgn2, therest, tbl_name, date_rem) (SELECT FBGN_GENE1_BD, FBGN_GENE2_AD, GENE1_INTERACTIONS_AS_BD || ', '|| GENE1_INTERACTIONS_TOTAL || ', '|| GENE2_INTERACTIONS_AS_AD || ', '|| GENE2_INTERACTIONS_TOTAL || ', '|| SCREEN || ', '|| REFERENCE || ', '|| DATE1 || ', '|| IST || ', '|| ISTS_RFCS || ', '|| DATA_VERSION || ', '|| DATE_LAST_UPDATED || ', '|| PBSSCORE as therest, upper('hybrigenics yth'), SYSDATE FROM hybrigenics_yth, tmp_holder WHERE FBGN_GENE1_BD = fbgn1_old AND FBGN_GENE2_AD = fbgn2_old);
DELETE from hybrigenics_yth where (FBGN_GENE1_BD, FBGN_GENE2_AD) in (SELECT fbgn1_old, fbgn2_old FROM tmp_holder);
INSERT INTO tbl_man_curate (fbgn1, fbgn2, therest, tbl_name, date_rem) (SELECT FBGN_GENE1_BD, FBGN_GENE2_AD, GENE1_INTERACTIONS_AS_BD || ', '|| GENE1_INTERACTIONS_TOTAL || ', '|| GENE2_INTERACTIONS_AS_AD || ', '|| GENE2_INTERACTIONS_TOTAL || ', '|| SCREEN || ', '|| REFERENCE || ', '|| DATE1 || ', '|| IST || ', '|| ISTS_RFCS || ', '|| DATA_VERSION || ', '|| DATE_LAST_UPDATED || ', '|| PBSSCORE as therest, upper('hybrigenics yth'), SYSDATE FROM hybrigenics_yth, not_mappable WHERE FBGN_GENE1_BD = secfbgn OR FBGN_GENE2_AD = secfbgn);
DELETE from hybrigenics_yth where ((FBGN_GENE1_BD) in (SELECT secfbgn FROM not_mappable) OR (FBGN_GENE2_AD) in (SELECT secfbgn FROM not_mappable));
COMMIT;
END;
```

```
create or replace PROCEDURE ins_corr
IS
BEGIN
        /* other physical*/
        INSERT INTO correlation (fly_gene1, fly_gene2) (SELECT DISTINCT t.Fly_gene1, t.Fly_gene2 FROM fly_other_physical t LEFT OUTER JOIN correlation c ON t.Fly_gene2 = c.Fly_gene2 AND t.Fly_gene1 = c.fly_gene1 WHERE c.fly_gene1 Is Null AND c.fly_gene2 Is Null);
        /* genetic interactions */
        INSERT INTO correlation (fly_gene1, fly_gene2) (SELECT DISTINCT t.Fly_gene1, t.Fly_gene2 FROM fly_genetic_interactions t LEFT OUTER JOIN correlation c ON t.Fly_gene2 = c.Fly_gene2 AND t.Fly_gene1 = c.fly_gene1 WHERE c.fly_gene1 Is Null AND c.fly_gene2 Is Null);
        /* human interologs */
        INSERT INTO correlation (fly_gene1, fly_gene2) (SELECT DISTINCT t.Fly_gene1, t.Fly_gene2 FROM human_interologs t LEFT OUTER JOIN correlation c ON t.Fly_gene2 = c.Fly_gene2 AND t.Fly_gene1 = c.fly_gene1 WHERE c.fly_gene1 Is Null AND c.fly_gene2 Is Null);
```

```
		/* yeast interologs */
		INSERT INTO correlation (fly_gene1, fly_gene2) (SELECT DISTINCT t.Fly_gene1, t.Fly_gene2 FROM yeast_interologs t LEFT OUTER JOIN correlation c ON t.Fly_gene2 = c.Fly_gene2 AND t.Fly_gene1 = c.fly_gene1 WHERE c.fly_gene1 Is Null AND c.fly_gene2 Is Null);
		/* worm interologs */
		INSERT INTO correlation (fly_gene1, fly_gene2) (SELECT DISTINCT t.Fly_gene1, t.Fly_gene2 FROM worm_interologs t LEFT OUTER JOIN correlation c ON t.Fly_gene2 = c.Fly_gene2 AND t.Fly_gene1 = c.fly_gene1 WHERE c.fly_gene1 Is Null AND c.fly_gene2 Is Null);
		/* tf-gene */
		INSERT INTO correlation (fly_gene1, fly_gene2) (SELECT DISTINCT t.Fly_tf_gene, t.Fly_target_gene FROM tf_gene t LEFT OUTER JOIN correlation c ON t.Fly_target_gene = c.Fly_gene2 AND t.Fly_tf_gene = c.fly_gene1 WHERE c.fly_gene1 Is Null AND c.fly_gene2 Is Null);
		/* rna-gene */
		INSERT INTO correlation (fly_gene1, fly_gene2) (SELECT DISTINCT t.Fly_rna_gene, t.Fly_target_gene FROM rna_gene t LEFT OUTER JOIN correlation c ON t.Fly_target_gene = c.Fly_gene2 AND t.Fly_rna_gene = c.fly_gene1 WHERE c.fly_gene1 Is Null AND c.fly_gene2 Is Null);
		/* curagen yth */
		INSERT INTO correlation (fly_gene1, fly_gene2) (SELECT DISTINCT t.Fbgn_gene1_bd, t.Fbgn_gene2_ad FROM curagen_yth t LEFT OUTER JOIN correlation c ON t.Fbgn_gene2_ad = c.Fly_gene2 AND t.Fbgn_gene1_bd = c.fly_gene1 WHERE c.fly_gene1 Is Null AND c.fly_gene2 Is Null);
		/* finley yth */
		INSERT INTO correlation (fly_gene1, fly_gene2) (SELECT DISTINCT t.Fbgn_gene1_bd, t.Fbgn_gene2_ad FROM finley_yth t LEFT OUTER JOIN correlation c ON t.Fbgn_gene2_ad = c.Fly_gene2 AND t.Fbgn_gene1_bd = c.fly_gene1 WHERE c.fly_gene1 Is Null AND c.fly_gene2 Is Null);
		/* hybrigenics yth */
		INSERT INTO correlation (fly_gene1, fly_gene2) (SELECT DISTINCT t.Fbgn_gene1_bd, t.Fbgn_gene2_ad FROM hybrigenics_yth t LEFT OUTER JOIN correlation c ON t.Fbgn_gene2_ad = c.Fly_gene2 AND t.Fbgn_gene1_bd = c.fly_gene1 WHERE c.fly_gene1 Is Null AND c.fly_gene2 Is Null);
		/* friedmanperrimon coap */
		INSERT INTO correlation (fly_gene1, fly_gene2) (SELECT DISTINCT t.Fbgn_bait, t.Fbgn_interactor FROM friedmanperrimon_coap t LEFT OUTER JOIN correlation c ON t.Fbgn_interactor = c.Fly_gene2 AND t.Fbgn_bait = c.fly_gene1 WHERE c.fly_gene1 Is Null AND c.fly_gene2 Is Null);
		/* dpim coapcomplex */
		INSERT INTO correlation (fly_gene1, fly_gene2) (SELECT DISTINCT t.Fbgn_bait, t.Fbgn_interactor FROM dpim_coapcomplex t LEFT OUTER JOIN correlation c ON t.Fbgn_interactor = c.Fly_gene2 AND t.Fbgn_bait = c.fly_gene1 WHERE c.fly_gene1 Is Null AND c.fly_gene2 Is Null);
COMMIT;
END;


create or replace PROCEDURE upd_fbgn_mapping
IS
BEGIN
		/*update new primary fbgns*/
		/* human interologs*/
		INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fly_gene1, fly_gene2, upper('human interologs col fbgn2'), SYSDATE from human_interologs inner join MAP_FBGN_SEC_PRI on human_interologs.FLY_GENE2 = map_fbgn_sec_pri.secfbgn);
		UPDATE human_interologs set fly_gene2 = (select prifbgn from map_fbgn_sec_pri where human_interologs.fly_gene2 = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where human_interologs.fly_gene2 = map_fbgn_sec_pri.secfbgn);
		INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fly_gene1, fly_gene2, upper('human interologs col fbgn1'), SYSDATE from human_interologs inner join MAP_FBGN_SEC_PRI on human_interologs.FLY_GENE1 = map_fbgn_sec_pri.secfbgn);
		UPDATE human_interologs set fly_gene1 = (select prifbgn from map_fbgn_sec_pri where human_interologs.fly_gene1 = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where human_interologs.fly_gene1 = map_fbgn_sec_pri.secfbgn);
		/*correlations*/
```

```
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fly_gene1, fly_gene2, upper('correlation col fbgn2'), SYSDATE from correlation inner join MAP_FBGN_SEC_PRI on correlation.FLY_GENE2 = map_fbgn_sec_pri.secfbgn);
UPDATE correlation set fly_gene2 = (select prifbgn from map_fbgn_sec_pri where correlation.FLY_GENE2 = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where correlation.FLY_GENE2 = map_fbgn_sec_pri.secfbgn);
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fly_gene1, fly_gene2, upper('correlation col fbgn1'), SYSDATE from correlation inner join MAP_FBGN_SEC_PRI on correlation.FLY_GENE1 = map_fbgn_sec_pri.secfbgn);
UPDATE correlation set fly_gene1 = (select prifbgn from map_fbgn_sec_pri where correlation.fly_gene1 = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where correlation.fly_gene1 = map_fbgn_sec_pri.secfbgn);
/*tf-gene*/
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fly_tf_gene, fly_target_gene, upper('tf gene col fbgn2'), SYSDATE from tf_gene inner join MAP_FBGN_SEC_PRI on tf_gene.FLY_target_GENE = map_fbgn_sec_pri.secfbgn);
UPDATE tf_gene set FLY_target_GENE = (select prifbgn from map_fbgn_sec_pri where tf_gene.FLY_target_GENE = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where tf_gene.FLY_target_GENE = map_fbgn_sec_pri.secfbgn);
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fly_tf_gene, fly_target_gene, upper('tf gene col fbgn1'), SYSDATE from tf_gene inner join MAP_FBGN_SEC_PRI on tf_gene.FLY_tf_GENE = map_fbgn_sec_pri.secfbgn);
UPDATE tf_gene set fly_tf_gene = (select prifbgn from map_fbgn_sec_pri where tf_gene.fly_tf_gene = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where tf_gene.fly_tf_gene = map_fbgn_sec_pri.secfbgn);
/* rna-gene*/
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fly_rna_gene, fly_target_gene, upper('rna gene col fbgn2'), SYSDATE from rna_gene inner join MAP_FBGN_SEC_PRI on rna_gene.FLY_target_GENE = map_fbgn_sec_pri.secfbgn);
UPDATE rna_gene set FLY_target_GENE = (select prifbgn from map_fbgn_sec_pri where rna_gene.FLY_target_GENE = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where rna_gene.FLY_target_GENE = map_fbgn_sec_pri.secfbgn);
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fly_rna_gene, fly_target_gene, upper('rna gene col fbgn1'), SYSDATE from rna_gene inner join MAP_FBGN_SEC_PRI on rna_gene.FLY_rna_GENE = map_fbgn_sec_pri.secfbgn);
UPDATE rna_gene set fly_rna_gene = (select prifbgn from map_fbgn_sec_pri where rna_gene.fly_rna_gene = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where rna_gene.fly_rna_gene = map_fbgn_sec_pri.secfbgn);
/*finley yth*/
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select FBGN_GENE1_BD, FBGN_GENE2_AD, upper('finley yth col fbgn2'), SYSDATE from finley_yth inner join MAP_FBGN_SEC_PRI on finley_yth.FBGN_GENE2_AD = map_fbgn_sec_pri.secfbgn);
UPDATE finley_yth set FBGN_GENE2_AD = (select prifbgn from map_fbgn_sec_pri where finley_yth.FBGN_GENE2_AD = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where finley_yth.FBGN_GENE2_AD = map_fbgn_sec_pri.secfbgn);
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select FBGN_GENE1_BD, FBGN_GENE2_AD, upper('finley yth col fbgn1'), SYSDATE from finley_yth inner join MAP_FBGN_SEC_PRI on finley_yth.FBGN_GENE1_BD = map_fbgn_sec_pri.secfbgn);
UPDATE finley_yth set FBGN_GENE1_BD = (select prifbgn from map_fbgn_sec_pri where finley_yth.FBGN_GENE1_BD = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where finley_yth.FBGN_GENE1_BD = map_fbgn_sec_pri.secfbgn);
/*curagen yth*/
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select FBGN_GENE1_BD, FBGN_GENE2_AD, upper('curagen yth col fbgn2'), SYSDATE from curagen_yth inner join MAP_FBGN_SEC_PRI on curagen_yth.FBGN_GENE2_AD = map_fbgn_sec_pri.secfbgn);
UPDATE curagen_yth set FBGN_GENE2_AD = (select prifbgn from map_fbgn_sec_pri where curagen_yth.FBGN_GENE2_AD = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where curagen_yth.FBGN_GENE2_AD = map_fbgn_sec_pri.secfbgn);
```

```
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select FBGN_GENE1_BD, FBGN_GENE2_AD, upper('curagen yth col fbgn1'), SYSDATE from curagen_yth inner join MAP_FBGN_SEC_PRI on curagen_yth.FBGN_GENE1_BD = map_fbgn_sec_pri.secfbgn);
UPDATE curagen_yth set FBGN_GENE1_BD = (select prifbgn from map_fbgn_sec_pri where curagen_yth.FBGN_GENE1_BD = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where curagen_yth.FBGN_GENE1_BD = map_fbgn_sec_pri.secfbgn);
/*hybrigenics yth*/
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select FBGN_GENE1_BD, FBGN_GENE2_AD, upper('hybrigenics yth col fbgn2'), SYSDATE from hybrigenics_yth inner join MAP_FBGN_SEC_PRI on hybrigenics_yth.FBGN_GENE2_AD = map_fbgn_sec_pri.secfbgn);
UPDATE hybrigenics_yth set FBGN_GENE2_AD = (select prifbgn from map_fbgn_sec_pri where hybrigenics_yth.FBGN_GENE2_AD = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where hybrigenics_yth.FBGN_GENE2_AD = map_fbgn_sec_pri.secfbgn);
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select FBGN_GENE1_BD, FBGN_GENE2_AD, upper('hybrigenics yth col fbgn1'), SYSDATE from hybrigenics_yth inner join MAP_FBGN_SEC_PRI on hybrigenics_yth.FBGN_GENE1_BD = map_fbgn_sec_pri.secfbgn);
UPDATE hybrigenics_yth set FBGN_GENE1_BD = (select prifbgn from map_fbgn_sec_pri where hybrigenics_yth.FBGN_GENE1_BD = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where hybrigenics_yth.FBGN_GENE1_BD = map_fbgn_sec_pri.secfbgn);
/*other physical*/
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fly_gene1, fly_gene2, upper('fly other physical col fbgn2'), SYSDATE from fly_other_physical inner join MAP_FBGN_SEC_PRI on fly_other_physical.FLY_GENE2 = map_fbgn_sec_pri.secfbgn);
UPDATE fly_other_physical set FLY_GENE2 = (select prifbgn from map_fbgn_sec_pri where fly_other_physical.FLY_GENE2 = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where fly_other_physical.FLY_GENE2 = map_fbgn_sec_pri.secfbgn);
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fly_gene1, fly_gene2, upper('fly other physical col fbgn1'), SYSDATE from fly_other_physical inner join MAP_FBGN_SEC_PRI on fly_other_physical.FLY_GENE1 = map_fbgn_sec_pri.secfbgn);
UPDATE fly_other_physical set fly_gene1 = (select prifbgn from map_fbgn_sec_pri where fly_other_physical.fly_gene1 = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where fly_other_physical.fly_gene1 = map_fbgn_sec_pri.secfbgn);
/* worm interologs*/
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fly_gene1, fly_gene2, upper('worm interologs col fbgn2'), SYSDATE from worm_interologs inner join MAP_FBGN_SEC_PRI on worm_interologs.FLY_GENE2 = map_fbgn_sec_pri.secfbgn);
UPDATE worm_interologs set FLY_GENE2 = (select prifbgn from map_fbgn_sec_pri where worm_interologs.FLY_GENE2 = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where worm_interologs.FLY_GENE2 = map_fbgn_sec_pri.secfbgn);
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fly_gene1, fly_gene2, upper('worm interologs col fbgn1'), SYSDATE from worm_interologs inner join MAP_FBGN_SEC_PRI on worm_interologs.FLY_GENE1 = map_fbgn_sec_pri.secfbgn);
UPDATE worm_interologs set fly_gene1 = (select prifbgn from map_fbgn_sec_pri where worm_interologs.fly_gene1 = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where worm_interologs.fly_gene1 = map_fbgn_sec_pri.secfbgn);
/*yeast interologs*/
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fly_gene1, fly_gene2, upper('yeast interologs col fbgn2'), SYSDATE from yeast_interologs inner join MAP_FBGN_SEC_PRI on yeast_interologs.FLY_GENE2 = map_fbgn_sec_pri.secfbgn);
UPDATE yeast_interologs set FLY_GENE2 = (select prifbgn from map_fbgn_sec_pri where yeast_interologs.FLY_GENE2 = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where yeast_interologs.FLY_GENE2 = map_fbgn_sec_pri.secfbgn);
INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fly_gene1, fly_gene2, upper('yeast interologs col fbgn1'), SYSDATE from yeast_interologs inner join MAP_FBGN_SEC_PRI on yeast_interologs.FLY_GENE1 = map_fbgn_sec_pri.secfbgn);
UPDATE yeast_interologs set fly_gene1 = (select prifbgn from map_fbgn_sec_pri where yeast_interologs.fly_gene1 = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where yeast_interologs.fly_gene1 = map_fbgn_sec_pri.secfbgn);
/*friedmanperrimon coap*/
```

```
		INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fbgn_bait, fbgn_interactor, upper('friedmanperrimon coap col fbgn2'), SYSDATE from friedmanperrimon_coap inner join MAP_FBGN_SEC_PRI on friedmanperrimon_coap.Fbgn_interactor = map_fbgn_sec_pri.secfbgn);
		UPDATE friedmanperrimon_coap set Fbgn_interactor = (select prifbgn from map_fbgn_sec_pri where friedmanperrimon_coap.Fbgn_interactor = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where friedmanperrimon_coap.Fbgn_interactor = map_fbgn_sec_pri.secfbgn);
		INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fbgn_bait, fbgn_interactor, upper('friedmanperrimon coap col fbgn1'), SYSDATE from friedmanperrimon_coap inner join MAP_FBGN_SEC_PRI on friedmanperrimon_coap.Fbgn_bait = map_fbgn_sec_pri.secfbgn);
		UPDATE friedmanperrimon_coap set fbgn_bait = (select prifbgn from map_fbgn_sec_pri where friedmanperrimon_coap.fbgn_bait = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where friedmanperrimon_coap.fbgn_bait = map_fbgn_sec_pri.secfbgn);
/*dpim coapcomplex*/
		INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fbgn_bait, fbgn_interactor, upper('dpim coapcomplex col fbgn2'), SYSDATE from dpim_coapcomplex inner join MAP_FBGN_SEC_PRI on dpim_coapcomplex.Fbgn_interactor = map_fbgn_sec_pri.secfbgn);
		UPDATE dpim_coapcomplex set Fbgn_interactor = (select prifbgn from map_fbgn_sec_pri where dpim_coapcomplex.Fbgn_interactor = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where dpim_coapcomplex.Fbgn_interactor = map_fbgn_sec_pri.secfbgn);
		INSERT INTO tbl_updated (fbgn1, fbgn2, tbl_name_col, date_upd) (select fbgn_bait, fbgn_interactor, upper('dpim coapcomplex col fbgn1'), SYSDATE from dpim_coapcomplex inner join MAP_FBGN_SEC_PRI on dpim_coapcomplex.Fbgn_bait = map_fbgn_sec_pri.secfbgn);
		UPDATE dpim_coapcomplex set fbgn_bait = (select prifbgn from map_fbgn_sec_pri where dpim_coapcomplex.fbgn_bait = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where dpim_coapcomplex.fbgn_bait = map_fbgn_sec_pri.secfbgn);
COMMIT;
END;



create or replace PROCEDURE upd_syms
IS
BEGIN
		/*update symbols*/
		/* human interologs*/
		UPDATE human_interologs set symbol_ad = (select symbol from fly_gene_attr where human_interologs.FLY_GENE2 = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where human_interologs.FLY_GENE2 = fly_gene_attr.fly_gene);
		UPDATE human_interologs set symbol_bd = (select symbol from fly_gene_attr where human_interologs.fly_gene1 = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where human_interologs.fly_gene1 = fly_gene_attr.fly_gene);
		/*correlations*/
		UPDATE correlation set symbol2 = (select symbol from fly_gene_attr where correlation.FLY_GENE2 = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where correlation.FLY_GENE2 = fly_gene_attr.fly_gene);
		UPDATE correlation set symbol1 = (select symbol from fly_gene_attr where correlation.fly_gene1 = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where correlation.fly_gene1 = fly_gene_attr.fly_gene);
		/*tf-gene*/
		UPDATE tf_gene set GENE_SYMBOL = (select symbol from fly_gene_attr where tf_gene.FLY_target_GENE = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where tf_gene.FLY_target_GENE = fly_gene_attr.fly_gene);
		UPDATE tf_gene set TF_SYMBOL = (select symbol from fly_gene_attr where tf_gene.fly_tf_gene = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where tf_gene.fly_tf_gene = fly_gene_attr.fly_gene);
		/*rna-gene*/
		UPDATE rna_gene set GENE_SYMBOL = (select symbol from fly_gene_attr where rna_gene.FLY_target_GENE = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where rna_gene.FLY_target_GENE = fly_gene_attr.fly_gene);
		UPDATE rna_gene set rna_SYMBOL = (select symbol from fly_gene_attr where rna_gene.fly_rna_gene = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where rna_gene.fly_rna_gene = fly_gene_attr.fly_gene);
```

```
/*finley yth*/
UPDATE finley_yth set symbol_ad = (select symbol from fly_gene_attr where finley_yth.FBGN_GENE2_AD = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where finley_yth.FBGN_GENE2_AD = fly_gene_attr.fly_gene);
UPDATE finley_yth set symbol_bd = (select symbol from fly_gene_attr where finley_yth.FBGN_GENE1_BD = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where finley_yth.FBGN_GENE1_BD = fly_gene_attr.fly_gene);
/*curagen yth*/
UPDATE curagen_yth set symbol_ad = (select symbol from fly_gene_attr where curagen_yth.FBGN_GENE2_AD = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where curagen_yth.FBGN_GENE2_AD = fly_gene_attr.fly_gene);
UPDATE curagen_yth set symbol_bd = (select symbol from fly_gene_attr where curagen_yth.FBGN_GENE1_BD = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where curagen_yth.FBGN_GENE1_BD = fly_gene_attr.fly_gene);
/*hybrigenics yth*/
UPDATE hybrigenics_yth set symbol_ad = (select symbol from fly_gene_attr where hybrigenics_yth.FBGN_GENE2_AD = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where hybrigenics_yth.FBGN_GENE2_AD = fly_gene_attr.fly_gene);
UPDATE hybrigenics_yth set symbol_bd = (select symbol from fly_gene_attr where hybrigenics_yth.FBGN_GENE1_BD = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where hybrigenics_yth.FBGN_GENE1_BD = fly_gene_attr.fly_gene);
/*other physical*/
UPDATE fly_other_physical set symbol_ad = (select symbol from fly_gene_attr where fly_other_physical.FLY_GENE2 = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where fly_other_physical.FLY_GENE2 = fly_gene_attr.fly_gene);
UPDATE fly_other_physical set symbol_bd = (select symbol from fly_gene_attr where fly_other_physical.fly_gene1 = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where fly_other_physical.fly_gene1 = fly_gene_attr.fly_gene);
/* worm interologs*/
UPDATE worm_interologs set symbol_ad = (select symbol from fly_gene_attr where worm_interologs.FLY_GENE2 = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where worm_interologs.FLY_GENE2 = fly_gene_attr.fly_gene);
UPDATE worm_interologs set symbol_bd = (select symbol from fly_gene_attr where worm_interologs.fly_gene1 = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where worm_interologs.fly_gene1 = fly_gene_attr.fly_gene);
/*yeast interologs*/
UPDATE yeast_interologs set symbol_ad = (select symbol from fly_gene_attr where yeast_interologs.FLY_GENE2 = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where yeast_interologs.FLY_GENE2 = fly_gene_attr.fly_gene);
UPDATE yeast_interologs set symbol_bd = (select symbol from fly_gene_attr where yeast_interologs.fly_gene1 = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where yeast_interologs.fly_gene1 = fly_gene_attr.fly_gene);
/*friedmanperrimon coap*/
UPDATE friedmanperrimon_coap set symbol_interactor = (select symbol from fly_gene_attr where friedmanperrimon_coap.Fbgn_interactor = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where friedmanperrimon_coap.Fbgn_interactor = fly_gene_attr.fly_gene);
UPDATE friedmanperrimon_coap set symbol_bait = (select symbol from fly_gene_attr where friedmanperrimon_coap.fbgn_bait = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where friedmanperrimon_coap.fbgn_bait = fly_gene_attr.fly_gene);
/*dpim coapcomplex */
UPDATE dpim_coapcomplex set symbol_interactor = (select symbol from fly_gene_attr where dpim_coapcomplex.Fbgn_interactor = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where dpim_coapcomplex.Fbgn_interactor = fly_gene_attr.fly_gene);
UPDATE dpim_coapcomplex set symbol_bait = (select symbol from fly_gene_attr where dpim_coapcomplex.fbgn_bait = fly_gene_attr.fly_gene) where exists (select symbol from fly_gene_attr where dpim_coapcomplex.fbgn_bait = fly_gene_attr.fly_gene);
COMMIT;
END;
```

```
create or replace PROCEDURE clean_tmp
IS
BEGIN
/* 1. copy all interaction fbgns into tmp table with 4 cols fbgn1_old, fbgn1_new, fbgn2_old, fbgn2_new - should be done when this proc is called*/
/* clean up tmp_discard */
DELETE FROM tmp_discard;
                /* 2. update fbgns to new pris */
UPDATE tmp_holder set fbgn1_new = (select prifbgn from map_fbgn_sec_pri where tmp_holder.fbgn1_new = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where tmp_holder.fbgn1_new = map_fbgn_sec_pri.secfbgn);
UPDATE tmp_holder set fbgn2_new = (select prifbgn from map_fbgn_sec_pri where tmp_holder.fbgn2_new = map_fbgn_sec_pri.secfbgn) where exists (select prifbgn from map_fbgn_sec_pri where tmp_holder.fbgn2_new = map_fbgn_sec_pri.secfbgn);
/* 3.Then do a group by on fbgn_new and get interactions that occur more than once. */
INSERT INTO tmp_discard (fbgn1_new, fbgn2_new) (SELECT tmp_holder.fbgn1_new, tmp_holder.fbgn2_new FROM tmp_holder GROUP BY tmp_holder.fbgn2_new, tmp_holder.fbgn1_new HAVING count(*)>1);
/* 4.          Next link on the new fbgns betw tmpholder and tmpdiscard and remove ones in tmpholder that don?t have interaction recs in tmpdiscard */
DELETE from tmp_holder where not (fbgn1_new, fbgn2_new) in (SELECT fbgn1_new, fbgn2_new FROM tmp_discard);
COMMIT;
END;
```

REFERENCES

1. Welch, G.R. and J.S. Clegg, *From protoplasmic theory to cellular systems biology: a 150-year reflection.* Am J Physiol Cell Physiol, 2010. **298**(6): p. C1280-90.
2. Hartwell, L.H., et al., *From molecular to modular cell biology.* Nature, 1999. **402**(6761 Suppl): p. C47-52.
3. Dunham, I., et al., *An integrated encyclopedia of DNA elements in the human genome.* Nature, 2012. **489**(7414): p. 57-74.
4. Graveley, B.R., et al., *The developmental transcriptome of Drosophila melanogaster.* Nature, 2011. **471**(7339): p. 473-9.
5. Cavalli, G. and T. Misteli, *Functional implications of genome topology.* Nat Struct Mol Biol, 2013. **20**(3): p. 290-9.
6. Misteli, T., *The cell biology of genomes: bringing the double helix to life.* Cell, 2013. **152**(6): p. 1209-12.
7. Chintapalli, V.R., J. Wang, and J.A. Dow, *Using FlyAtlas to identify better Drosophila melanogaster models of human disease.* Nat Genet, 2007. **39**(6): p. 715-20.
8. Bhalla, U.S. and R. Iyengar, *Emergent properties of networks of biological signaling pathways.* Science, 1999. **283**(5400): p. 381-7.
9. Trewavas, A., *A brief history of systems biology. "Every object that biology studies is a system of systems." Francois Jacob (1974).* Plant Cell, 2006. **18**(10): p. 2420-30.

10. Alon, U., *Biological networks: the tinkerer as an engineer.* Science, 2003. **301**(5641): p. 1866-7.

11. Westerhoff, H.V. and B.O. Palsson, *The evolution of molecular biology into systems biology.* Nat Biotechnol, 2004. **22**(10): p. 1249-52.

12. Kirschner, M.W., *The meaning of systems biology.* Cell, 2005. **121**(4): p. 503-4.

13. Brent, R., *Genomic biology.* Cell, 2000. **100**(1): p. 169-83.

14. Baker, M., *Proteomics: The interaction map.* Nature, 2012. **484**(7393): p. 271-5.

15. Arnosti, D.N. and A. Ay, *Boolean modeling of gene regulatory networks: Driesch redux.* Proc Natl Acad Sci U S A, 2012. **109**(45): p. 18239-40.

16. Uetz, P. and R.L. Finley, Jr., *From protein networks to biological systems.* FEBS Lett, 2005. **579**(8): p. 1821-7.

17. Barabasi, A.L. and Z.N. Oltvai, *Network biology: understanding the cell's functional organization.* Nat Rev Genet, 2004. **5**(2): p. 101-13.

18. Bonn, S. and E.E. Furlong, *cis-Regulatory networks during development: a view of Drosophila.* Curr Opin Genet Dev, 2008. **18**(6): p. 513-20.

19. Chuang, H.Y., M. Hofree, and T. Ideker, *A decade of systems biology.* Annu Rev Cell Dev Biol, 2010. **26**: p. 721-44.

20. Brent, R. and J. Bruck, *2020 computing: can computers help to explain biology?* Nature, 2006. **440**(7083): p. 416-7.

21. Ideker, T., T. Galitski, and L. Hood, *A new approach to decoding life: systems biology.* Annu Rev Genomics Hum Genet, 2001. **2**: p. 343-72.

22. Kitano, H., *Computational systems biology.* Nature, 2002. **420**(6912): p. 206-10.

23. Lazebnik, Y., *Can a biologist fix a radio?--Or, what I learned while studying apoptosis.* Cancer Cell, 2002. **2**(3): p. 179-82.

24. Braun, P. and A.C. Gingras, *History of protein-protein interactions: from egg-white to complex networks.* Proteomics, 2012. **12**(10): p. 1478-98.

25. Cowley, A.W., Jr., *The elusive field of systems biology.* Physiol Genomics, 2004. **16**(3): p. 285-6.

26. Mesarovic, M.D., S.N. Sreenath, and J.D. Keene, *Search for organising principles: understanding in systems biology.* Syst Biol (Stevenage), 2004. **1**(1): p. 19-27.

27. Stanyon, C.A. and R.L. Finley, Jr., *Progress and potential of Drosophila protein interaction maps.* Pharmacogenomics, 2000. **1**(4): p. 417-31.

28. Tarassov, K., et al., *An in vivo map of the yeast protein interactome.* Science, 2008. **320**(5882): p. 1465-70.

29. Babu, M., et al., *Interaction landscape of membrane-protein complexes in Saccharomyces cerevisiae.* Nature, 2012. **489**(7417): p. 585-9.

30. Yu, H., et al., *High-quality binary protein interaction map of the yeast interactome network.* Science, 2008. **322**(5898): p. 104-10.

31. Gavin, A.C., et al., *Proteome survey reveals modularity of the yeast cell machinery.* Nature, 2006. **440**(7084): p. 631-6.

32. Krogan, N.J., et al., *Global landscape of protein complexes in the yeast Saccharomyces cerevisiae.* Nature, 2006. **440**(7084): p. 637-43.

33. Li, S., et al., *A map of the interactome network of the metazoan C. elegans.* Science, 2004. **303**(5657): p. 540-3.

34. Formstecher, E., et al., *Protein interaction mapping: a Drosophila case study.* Genome Res, 2005. **15**(3): p. 376-84.

35. Giot, L., et al., *A protein interaction map of Drosophila melanogaster.* Science, 2003. **302**(5651): p. 1727-36.

36. Stanyon, C.A., et al., *A Drosophila protein-interaction map centered on cell-cycle regulators.* Genome Biol, 2004. **5**(12): p. R96.

37. Guruharsha, K.G., et al., *A protein complex network of Drosophila melanogaster.* Cell, 2011. **147**(3): p. 690-703.

38. Friedman, A.A., et al., *Proteomic and functional genomic landscape of receptor tyrosine kinase and ras to extracellular signal-regulated kinase signaling.* Sci Signal, 2011. **4**(196): p. rs10.

39. Stelzl, U., et al., *A human protein-protein interaction network: a resource for annotating the proteome.* Cell, 2005. **122**(6): p. 957-68.

40. Ewing, R.M., et al., *Large-scale mapping of human protein-protein interactions by mass spectrometry.* Mol Syst Biol, 2007. **3**: p. 89.

41. Rual, J.F., et al., *Towards a proteome-scale map of the human protein-protein interaction network.* Nature, 2005. **437**(7062): p. 1173-8.

42. Phizicky, E., et al., *Protein analysis on a proteomic scale.* Nature, 2003. **422**(6928): p. 208-15.

43. Gavin, A.C., et al., *Functional organization of the yeast proteome by systematic analysis of protein complexes.* Nature, 2002. **415**(6868): p. 141-7.

44. Havugimana, P.C., et al., *A census of human soluble protein complexes.* Cell, 2012. **150**(5): p. 1068-81.

45. Jensen, L.J. and P. Bork, *Biochemistry. Not comparable, but complementary.* Science, 2008. **322**(5898): p. 56-7.

46. Ceol, A., et al., *MINT, the molecular interaction database: 2009 update.* Nucleic Acids Res, 2010. **38**(Database issue): p. D532-9.

47. Chatr-Aryamontri, A., et al., *The BioGRID interaction database: 2013 update.* Nucleic Acids Res, 2012.

48. Kerrien, S., et al., *The IntAct molecular interaction database in 2012.* Nucleic Acids Res, 2012. **40**(Database issue): p. D841-6.

49. Spitz, F. and E.E. Furlong, *Transcription factors: from enhancer binding to developmental control.* Nat Rev Genet, 2012. **13**(9): p. 613-26.

50. Lenhard, B., A. Sandelin, and P. Carninci, *Metazoan promoters: emerging characteristics and insights into transcriptional regulation.* Nat Rev Genet, 2012. **13**(4): p. 233-45.

51. Roy, S., et al., *Identification of functional elements and regulatory circuits by Drosophila modENCODE.* Science, 2010. **330**(6012): p. 1787-97.

52. Gallo, S.M., et al., *REDfly v3.0: toward a comprehensive database of transcriptional regulatory elements in Drosophila.* Nucleic Acids Res, 2011. **39**(Database issue): p. D118-23.

53. Gerstein, M.B., et al., *Architecture of the human regulatory network derived from ENCODE data.* Nature, 2012. **489**(7414): p. 91-100.

54. Neph, S., et al., *Circuitry and dynamics of human transcription factor regulatory networks.* Cell, 2012. **150**(6): p. 1274-86.

55. Marbach, D., et al., *Predictive regulatory models in Drosophila melanogaster by integrative inference of transcriptional networks.* Genome Res, 2012. **22**(7): p. 1334-49.

56. Gross, D.S. and W.T. Garrard, *Nuclease hypersensitive sites in chromatin.* Annu Rev Biochem, 1988. **57**: p. 159-97.

57. Papadopoulos, G.L., et al., *The database of experimentally supported targets: a functional update of TarBase.* Nucleic Acids Res, 2009. **37**(Database issue): p. D155-8.

58. Bartel, D.P., *MicroRNAs: target recognition and regulatory functions.* Cell, 2009. **136**(2): p. 215-33.

59. Griffiths-Jones, S., et al., *miRBase: tools for microRNA genomics.* Nucleic Acids Res, 2008. **36**(Database issue): p. D154-8.

60. Lewis, B.P., C.B. Burge, and D.P. Bartel, *Conserved seed pairing, often flanked by adenosines, indicates that thousands of human genes are microRNA targets.* Cell, 2005. **120**(1): p. 15-20.

61. Schnall-Levin, M., et al., *Conserved microRNA targeting in Drosophila is as widespread in coding regions as in 3'UTRs.* Proc Natl Acad Sci U S A, 2010. **107**(36): p. 15751-6.

62. Emmert-Streib, F. and M. Dehmer, *Networks for systems biology: conceptual connection of data and function.* IET Syst Biol, 2011. **5**(3): p. 185-207.

63. Braun, P., *Interactome mapping for analysis of complex phenotypes: insights from benchmarking binary interaction assays.* Proteomics, 2012. **12**(10): p. 1499-518.

64. Reiners, J., et al., *Molecular basis of human Usher syndrome: deciphering the meshes of the Usher protein network provides insights into the pathomechanisms of the Usher disease.* Exp Eye Res, 2006. **83**(1): p. 97-119.

65. Kohler, S., et al., *Walking the interactome for prioritization of candidate disease genes.* Am J Hum Genet, 2008. **82**(4): p. 949-58.

66. Bromberg, Y., *Chapter 15: disease gene prioritization.* PLoS Comput Biol, 2013. **9**(4): p. e1002902.

67. Chen, J., et al., *ToppGene Suite for gene list enrichment analysis and candidate gene prioritization.* Nucleic Acids Res, 2009. **37**(Web Server issue): p. W305-11.

68. Magger, O., et al., *Enhancing the Prioritization of Disease-Causing Genes through Tissue Specific Protein Interaction Networks.* PLoS Comput Biol, 2012. **8**(9): p. e1002690.

69. Song, J. and M. Singh, *How and when should interactome-derived clusters be used to predict functional modules and protein function?* Bioinformatics, 2009. **25**(23): p. 3143-50.

70. Rachlin, J., et al., *Biological context networks: a mosaic view of the interactome.* Mol Syst Biol, 2006. **2**: p. 66.

71. Sorge, S., et al., *The cis-regulatory code of Hox function in Drosophila.* EMBO J, 2012. **31**(15): p. 3323-33.

72. Marinho, J., et al., *The nucleolar protein Viriato/Nol12 is required for the growth and differentiation progression activities of the Dpp pathway during Drosophila eye development.* Dev Biol, 2013. **377**(1): p. 154-65.

73. Nagel, A.C., et al., *Dorso-ventral axis formation of the Drosophila oocyte requires Cyclin G.* Hereditas, 2012. **149**(5): p. 186-96.

74. Nagel, A.C., et al., *Cyclin G is involved in meiotic recombination repair in Drosophila melanogaster.* J Cell Sci, 2012. **125**(Pt 22): p. 5555-63.

75. Haase Gilbert, E., et al., *Drosophila signal peptidase complex member spase12 is required for development and cell differentiation.* PLoS One, 2013. **8**(4): p. e60908.

76. Molnar, C., et al., *Genetic annotation of gain-of-function screens using RNA interference and in situ hybridization of candidate genes in the Drosophila wing.* Genetics, 2012. **192**(2): p. 741-52.

77. Yeger-Lotem, E., et al., *Bridging high-throughput genetic and transcriptional data reveals cellular responses to alpha-synuclein toxicity.* Nat Genet, 2009. **41**(3): p. 316-23.

78. Huang, S.S. and E. Fraenkel, *Integrating proteomic, transcriptional, and interactome data reveals hidden components of signaling and regulatory networks.* Sci Signal, 2009. **2**(81): p. ra40.

79. Ma'ayan, A., et al., *Formation of regulatory patterns during signal propagation in a Mammalian cellular network.* Science, 2005. **309**(5737): p. 1078-83.

80. Cristino, A.S., et al., *Neurodevelopmental and neuropsychiatric disorders represent an interconnected molecular system.* Mol Psychiatry, 2013.

81. Wang, L., et al., *Integrating multiple types of data to predict novel cell cycle-related genes.* BMC Syst Biol, 2011. **5 Suppl 1**: p. S9.

82. Papp, D., et al., *The NRF2-related interactome and regulome contain multifunctional proteins and fine-tuned autoregulatory loops.* FEBS Lett, 2012. **586**(13): p. 1795-802.

83. McQuilton, P., S.E. St Pierre, and J. Thurmond, *FlyBase 101--the basics of navigating FlyBase.* Nucleic Acids Res, 2012. **40**(Database issue): p. D706-14.

84. Yook, K., et al., *WormBase 2012: more genomes, more data, new website.* Nucleic Acids Res, 2012. **40**(Database issue): p. D735-41.

85. *Database resources of the National Center for Biotechnology Information.* Nucleic Acids Res, 2013. **41**(Database issue): p. D8-D20.

86. Leonelli, S. and R.A. Ankeny, *Re-thinking organisms: The impact of databases on model organism biology.* Stud Hist Philos Biol Biomed Sci, 2012. **43**(1): p. 29-36.

87. Yu, J., et al., *DroID: the Drosophila Interactions Database, a comprehensive resource for annotated gene and protein interactions.* BMC Genomics, 2008. **9**: p. 461.

88. *A user's guide to the encyclopedia of DNA elements (ENCODE).* PLoS Biol, 2011. **9**(4): p. e1001046.

89. Contrino, S., et al., *modMine: flexible access to modENCODE data.* Nucleic Acids Res, 2012. **40**(Database issue): p. D1082-8.

90. Friedman, R.C., et al., *Most mammalian mRNAs are conserved targets of microRNAs.* Genome Res, 2009. **19**(1): p. 92-105.

91. Enright, A.J., et al., *MicroRNA targets in Drosophila.* Genome Biol, 2003. **5**(1): p. R1.

92. Grun, D., et al., *microRNA target predictions across seven Drosophila species and comparison to mammalian targets.* PLoS Comput Biol, 2005. **1**(1): p. e13.

93. Barrett, T., et al., *NCBI GEO: archive for functional genomics data sets--update.* Nucleic Acids Res, 2013. **41**(Database issue): p. D991-5.

94. Schaab, C., et al., *Analysis of high accuracy, quantitative proteomics data in the MaxQB database.* Mol Cell Proteomics, 2012. **11**(3): p. M111 014068.

95. Joyce, A.R. and B.O. Palsson, *The model organism as a system: integrating 'omics' data sets.* Nat Rev Mol Cell Biol, 2006. **7**(3): p. 198-210.

96. Przytycka, T.M., M. Singh, and D.K. Slonim, *Toward the dynamic interactome: it's about time.* Brief Bioinform, 2010. **11**(1): p. 15-29.

97. Philippi, S. and J. Kohler, *Addressing the problems with life-science databases for traditional uses and systems biology.* Nat Rev Genet, 2006. **7**(6): p. 482-8.

98. Murali, T., et al., *DroID 2011: a comprehensive, integrated resource for protein, transcription factor, RNA and gene interactions for Drosophila.* Nucleic Acids Res, 2011. **39**(Database issue): p. D736-43.

99. Chatr-aryamontri, A., et al., *MINT: the Molecular INTeraction database.* Nucleic Acids Res, 2007. **35**(Database issue): p. D572-4.

100. Stark, C., et al., *BioGRID: a general repository for interaction datasets.* Nucleic Acids Res, 2006. **34**(Database issue): p. D535-9.

101. Aranda, B., et al., *The IntAct molecular interaction database in 2010.* Nucleic Acids Res, 2009.

102. Chen, J.Y., S. Mamidipalli, and T. Huan, *HAPPI: an online database of comprehensive human annotated and predicted protein interactions.* BMC Genomics, 2009. **10 Suppl 1**: p. S16.

103. Reguly, T., et al., *Comprehensive curation and analysis of global interaction networks in Saccharomyces cerevisiae.* J Biol, 2006. **5**(4): p. 11.

104. Pacifico, S., et al., *A database and tool, IM Browser, for exploring and integrating emerging gene and protein interaction data for Drosophila.* BMC Bioinformatics, 2006. **7**: p. 195.

105. Cline, M.S., et al., *Integration of biological networks and gene expression data using Cytoscape.* Nat Protoc, 2007. **2**(10): p. 2366-82.

106. Yu, J. and R.L. Finley, Jr., *Combining multiple positive training sets to generate confidence scores for protein-protein interactions.* Bioinformatics, 2009. **25**(1): p. 105-11.

107. Chikina, M.D., et al., *Global prediction of tissue-specific gene expression and context-dependent gene networks in Caenorhabditis elegans.* PLoS Comput Biol, 2009. **5**(6): p. e1000417.

108. Wilson, R.J., J.L. Goodman, and V.B. Strelets, *FlyBase: integration and improvements to query tools.* Nucleic Acids Res, 2008. **36**(Database issue): p. D588-93.

109. Mungall, C.J. and D.B. Emmert, *A Chado case study: an ontology-based modular schema for representing genome-associated biological information.* Bioinformatics, 2007. **23**(13): p. i337-46.

110. Consortium, G.O., *The Gene Ontology in 2010: extensions and refinements.* Nucleic Acids Res, 2010. **38**(Database issue): p. D331-5.

111. Tweedie, S., et al., *FlyBase: enhancing Drosophila Gene Ontology annotations.* Nucleic Acids Res, 2009. **37**(Database issue): p. D555-9.

112. Ostlund, G., et al., *InParanoid 7: new algorithms and tools for eukaryotic orthology analysis.* Nucleic Acids Res, 2010. **38**(Database issue): p. D196-203.

113. McKusick-Nathans Institute of Genetic Medicine, J.H.U.B., MD), World Wide Web URL: http://omim.org/, *Online Mendelian Inheritance in Man, OMIM®.* 2013.

114. Isserlin, R., R.A. El-Badrawi, and G.D. Bader, *The Biomolecular Interaction Network Database in PSI-MI 2.5.* Database (Oxford), 2011. **2011**: p. baq037.

115. Walhout, A.J., et al., *Protein interaction mapping in C. elegans using proteins involved in vulval development.* Science, 2000. **287**(5450): p. 116-22.

116. Matthews, L.R., et al., *Identification of potential interaction networks using sequence-based searches for conserved protein-protein interactions or "interologs"*. Genome Res, 2001. **11**(12): p. 2120-6.

117. Yu, H., et al., *Annotation transfer between genomes: protein-protein interologs and protein-DNA regulogs.* Genome Res, 2004. **14**(6): p. 1107-18.

118. Matthews, L., et al., *Reactome knowledgebase of human biological pathways and processes.* Nucleic Acids Res, 2009. **37**(Database issue): p. D619-22.

119. Prasad, T.S., K. Kandasamy, and A. Pandey, *Human Protein Reference Database and Human Proteinpedia as discovery tools for systems biology.* Methods Mol Biol, 2009. **577**: p. 67-79.

120. Lewis, A.C., et al., *What evidence is there for the homology of protein-protein interactions?* PLoS Comput Biol, 2012. **8**(9): p. e1002645.

121. Brown, K.R. and I. Jurisica, *Unequal evolutionary conservation of human protein interactions in interologous networks.* Genome Biol, 2007. **8**(5): p. R95.

122. Zinman, G.E., S. Zhong, and Z. Bar-Joseph, *Biological interaction networks are conserved at the module level.* BMC Syst Biol, 2011. **5**: p. 134.

123. Yu, J., T. Murali, and R.L. Finley, Jr., *Assigning confidence scores to protein-protein interactions.* Methods Mol Biol, 2012. **812**: p. 161-74.

124. Sharan, R., et al., *Conserved patterns of protein interaction in multiple species.* Proc Natl Acad Sci U S A, 2005. **102**(6): p. 1974-9.

125. Halfon, M.S., S.M. Gallo, and C.M. Bergman, *REDfly 2.0: an integrated database of cis-regulatory modules and transcription factor binding sites in Drosophila.* Nucleic Acids Res, 2008. **36**(Database issue): p. D594-8.

126. Ruby, J.G., et al., *Evolution, biogenesis, expression, and target predictions of a substantially expanded set of Drosophila microRNAs.* Genome Res, 2007. **17**(12): p. 1850-64.

127. Griffiths-Jones, S., *miRBase: microRNA sequences and annotation.* Curr Protoc Bioinformatics, 2010. **Chapter 12**: p. Unit 12 9 1-10.

128. Tomancak, P., et al., *Global analysis of patterns of gene expression during Drosophila embryogenesis.* Genome Biol, 2007. **8**(7): p. R145.

129. Barrett, T., et al., *NCBI GEO: archive for high-throughput functional genomic data.* Nucleic Acids Res, 2009. **37**(Database issue): p. D885-90.

130. Zhang, Q., et al., *A model-based method for gene dependency measurement.* PLoS One, 2012. **7**(7): p. e40918.

131. Haye, A., J. Albert, and M. Rooman, *Robust non-linear differential equation models of gene expression evolution across Drosophila development.* BMC Res Notes, 2012. **5**: p. 46.

132. Herrmann, C., et al., *i-cisTarget: an integrative genomics method for the prediction of regulatory features and cis-regulatory modules.* Nucleic Acids Res, 2012. **40**(15): p. e114.

133. Titsias, M.K., et al., *Identifying targets of multiple co-regulating transcription factors from expression time-series by Bayesian model comparison.* BMC Syst Biol, 2012. **6**: p. 53.

134. Hansen, M.E. and R.J. Kulathinal, *Sex-biased networks and nodes of sexually antagonistic conflict in Drosophila.* Int J Evol Biol, 2013. **2013**: p. 545392.

135. Zhou, S., et al., *Phenotypic plasticity of the Drosophila transcriptome.* PLoS Genet, 2012. **8**(3): p. e1002593.

136. De Arras, L., et al., *An evolutionarily conserved innate immunity protein interaction network.* J Biol Chem, 2013. **288**(3): p. 1967-78.

137. Ni, X., et al., *Adaptive evolution and the birth of CTCF binding sites in the Drosophila genome.* PLoS Biol, 2012. **10**(11): p. e1001420.

138. Sun, M.G. and P.M. Kim, *Evolution of biological interaction networks: from models to real data.* Genome Biol, 2011. **12**(12): p. 235.

139. Grubbs, N., et al., *New components of drosophila leg development identified through genome wide association studies.* PLoS One, 2013. **8**(4): p. e60261.

140. Renaud, Y., et al., *DroPNet: a web portal for integrated analysis of Drosophila protein-protein interaction networks.* Nucleic Acids Res, 2012. **40**(Web Server issue): p. W134-9.

141. Fernandez-Suarez, X.M. and M.Y. Galperin, *The 2013 Nucleic Acids Research Database Issue and the online molecular biology database collection.* Nucleic Acids Res, 2013. **41**(Database issue): p. D1-7.

142. Miteva, Y.V., H.G. Budayeva, and I.M. Cristea, *Proteomics-based methods for discovery, quantification, and validation of protein-protein interactions.* Anal Chem, 2013. **85**(2): p. 749-68.

143. Su, A.I., et al., *A gene atlas of the mouse and human protein-encoding transcriptomes.* Proc Natl Acad Sci U S A, 2004. **101**(16): p. 6062-7.

144. Ramskold, D., et al., *An abundance of ubiquitously expressed genes revealed by tissue transcriptome sequence data.* PLoS Comput Biol, 2009. **5**(12): p. e1000598.

145. Saito-Hisaminato, A., et al., *Genome-wide profiling of gene expression in 29 normal human tissues with a cDNA microarray.* DNA Res, 2002. **9**(2): p. 35-45.

146. Hooper, S.D., et al., *Identification of tightly regulated groups of genes during Drosophila melanogaster embryogenesis.* Mol Syst Biol, 2007. **3**: p. 72.

147. Tomancak, P., et al., *Systematic determination of patterns of gene expression during Drosophila embryogenesis.* Genome Biol, 2002. **3**(12): p. RESEARCH0088.

148. Deplancke, B., *Experimental advances in the characterization of metazoan gene regulatory networks.* Brief Funct Genomic Proteomic, 2009. **8**(1): p. 12-27.

149. Dow, J.A. and M.F. Romero, *Drosophila provides rapid modeling of renal development, function, and disease.* Am J Physiol Renal Physiol, 2010. **299**(6): p. F1237-44.

150. Bossi, A. and B. Lehner, *Tissue specificity and the human protein interaction network.* Mol Syst Biol, 2009. **5**: p. 260.

151. Emig, D. and M. Albrecht, *Tissue-specific proteins and functional implications.* J Proteome Res, 2011. **10**(4): p. 1893-903.

152. Lin, W.H., W.C. Liu, and M.J. Hwang, *Topological and organizational properties of the products of house-keeping and tissue-specific genes in protein-protein interaction networks.* BMC Syst Biol, 2009. **3**: p. 32.

153. Schaefer, M.H., et al., *Adding protein context to the human protein-protein interaction network to reveal meaningful interactions.* PLoS Comput Biol, 2013. **9**(1): p. e1002860.

154. Guan, Y., et al., *Tissue-specific functional networks for prioritizing phenotype and disease genes.* PLoS Comput Biol, 2012. **8**(9): p. e1002694.

155. Lopes, T.J., et al., *Tissue-specific subnetworks and characteristics of publicly available human protein interaction databases.* Bioinformatics, 2011. **27**(17): p. 2414-21.

156. Shlomi, T., et al., *Network-based prediction of human tissue-specific metabolism.* Nat Biotechnol, 2008. **26**(9): p. 1003-10.

157. Ott, J., *Analysis of Human Genetic Linkage*. 3 ed1999, Baltimore: The Johns Hopkins Univeristy Press. 382.

158. Snedecor, G.W. and W.G. Cochran, *Statistical Methods*. 8th ed1989, Ames: Iowa State Univeristy Press. 490.

159. Huang da, W., B.T. Sherman, and R.A. Lempicki, *Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources.* Nat Protoc, 2009. **4**(1): p. 44-57.

160. Weng, M.P. and B.Y. Liao, *DroPhEA: Drosophila phenotype enrichment analysis for insect functional genomics.* Bioinformatics, 2011. **27**(22): p. 3218-9.

161. Maere, S., K. Heymans, and M. Kuiper, *BiNGO: a Cytoscape plugin to assess overrepresentation of gene ontology categories in biological networks.* Bioinformatics, 2005. **21**(16): p. 3448-9.

162. Lehner, B. and A.G. Fraser, *Protein domains enriched in mammalian tissue-specific or widely expressed genes.* Trends Genet, 2004. **20**(10): p. 468-72.

163. Winter, E.E., L. Goodstadt, and C.P. Ponting, *Elevated rates of protein secretion, evolution, and disease among tissue-specific genes.* Genome Res, 2004. **14**(1): p. 54-61.

164. Zhang, L. and W.H. Li, *Mammalian housekeeping genes evolve more slowly than tissue-specific genes.* Mol Biol Evol, 2004. **21**(2): p. 236-9.

165. Freilich, S., et al., *Relationship between the tissue-specificity of mouse gene expression and the evolutionary origin and function of the proteins.* Genome Biol, 2005. **6**(7): p. R56.

166. Tu, Z., et al., *Further understanding human disease genes by comparing with housekeeping genes and other genes.* BMC Genomics, 2006. **7**: p. 31.

167. Maier, T., M. Guell, and L. Serrano, *Correlation of mRNA and protein in complex biological samples.* FEBS Lett, 2009. **583**(24): p. 3966-73.

168. Edgar, B.A. and C.F. Lehner, *Developmental control of cell cycle regulators: a fly's perspective.* Science, 1996. **274**(5293): p. 1646-52.

169. Chia, W., W.G. Somers, and H. Wang, *Drosophila neuroblast asymmetric divisions: cell cycle regulators, asymmetric protein localization, and tumorigenesis.* J Cell Biol, 2008. **180**(2): p. 267-72.

170. Hafer, N., et al., *The Drosophila CPEB protein Orb2 has a novel expression pattern and is important for asymmetric cell division and nervous system function.* Genetics, 2011. **189**(3): p. 907-21.

171. Weigmann, K., et al., *FlyMove--a new way to look at development of Drosophila.* Trends Genet, 2003. **19**(6): p. 310-1.

172. Kalinka, A.T., et al., *Gene expression divergence recapitulates the developmental hourglass model.* Nature, 2010. **468**(7325): p. 811-4.

173. Li, S., et al., *Gene-sharing networks reveal organizing principles of transcriptomes in Arabidopsis and other multicellular organisms.* Plant Cell, 2012. **24**(4): p. 1362-78.

174. Duffy, J.B., D.A. Harrison, and N. Perrimon, *Identifying loci required for follicular patterning using directed mosaics.* Development, 1998. **125**(12): p. 2263-71.

175. Scholl, F.A., et al., *Mek1/2 MAPK kinases are essential for Mammalian development, homeostasis, and Raf-induced hyperplasia.* Dev Cell, 2007. **12**(4): p. 615-29.

176. Weinkove, D., et al., *Regulation of imaginal disc cell size, cell number and organ size by Drosophila class I(A) phosphoinositide 3-kinase and its adaptor*. Curr Biol, 1999. **9**(18): p. 1019-29.

177. Weber, U., et al., *Novel regulators of planar cell polarity: a genetic analysis in Drosophila.* Genetics, 2012. **191**(1): p. 145-62.

178. Pak, C., et al., *Mutation of the conserved polyadenosine RNA binding protein, ZC3H14/dNab2, impairs neural function in Drosophila and humans.* Proc Natl Acad Sci U S A, 2011. **108**(30): p. 12390-5.

179. Peru, Y.C.d.P.R.L., et al., *Adult neuronal Arf6 controls ethanol-induced behavior with Arfaptin downstream of Rac1 and RhoGAP18B.* J Neurosci, 2012. **32**(49): p. 17706-13.

180. de Ligt, J., et al., *Diagnostic exome sequencing in persons with severe intellectual disability.* N Engl J Med, 2012. **367**(20): p. 1921-9.

181. McGurk, L. and N.M. Bonini, *Protein interacting with C kinase (PICK1) is a suppressor of spinocerebellar ataxia 3-associated neurodegeneration in Drosophila.* Hum Mol Genet, 2012. **21**(1): p. 76-84.

182. Duvall, L.B. and P.H. Taghert, *The circadian neuropeptide PDF signals preferentially through a specific adenylate cyclase isoform AC3 in M pacemakers of Drosophila.* PLoS Biol, 2012. **10**(6): p. e1001337.

183. Sepp, K.J., et al., *Identification of neural outgrowth genes using genome-wide RNAi.* PLoS Genet, 2008. **4**(7): p. e1000111.

184. Tan, Y., et al., *Gilgamesh is required for rutabaga-independent olfactory learning in Drosophila.* Neuron, 2010. **67**(5): p. 810-20.

185. Hummel, T., et al., *Temporal control of glial cell migration in the Drosophila eye requires gilgamesh, hedgehog, and eye specification genes.* Neuron, 2002. **33**(2): p. 193-203.

186. Chao, R., et al., *A male with unilateral microphthalmia reveals a role for TMX3 in eye development.* PLoS One, 2010. **5**(5): p. e10565.

187. Han, J., et al., *The fly CAMTA transcription factor potentiates deactivation of rhodopsin, a G protein-coupled light receptor.* Cell, 2006. **127**(4): p. 847-58.

188. Asztalos, Z., et al., *Protein phosphatase 1-deficient mutant Drosophila is affected in habituation and associative learning.* J Neurosci, 1993. **13**(3): p. 924-30.

189. Martin-Blanco, E., *p38 MAPK signalling cascades: ancient roles and new functions.* Bioessays, 2000. **22**(7): p. 637-45.

190. Wang, Y.J. and H.W. Brock, *Polyhomeotic stably associates with molecular chaperones Hsc4 and Droj2 in Drosophila Kc1 cells.* Dev Biol, 2003. **262**(2): p. 350-60.

191. Iwasaki, S., et al., *Hsc70/Hsp90 chaperone machinery mediates ATP-dependent RISC loading of small RNA duplexes.* Mol Cell, 2010. **39**(2): p. 292-9.

192. Uechi, T., et al., *Ribosomal protein gene knockdown causes developmental defects in zebrafish.* PLoS One, 2006. **1**: p. e37.

193. Raices, M. and M.A. D'Angelo, *Nuclear pore complex composition: a new regulator of tissue-specific and developmental functions.* Nat Rev Mol Cell Biol, 2012. **13**(11): p. 687-99.

194. Bowne, S.J., et al., *Why do mutations in the ubiquitously expressed housekeeping gene IMPDH1 cause retina-specific photoreceptor degeneration?* Invest Ophthalmol Vis Sci, 2006. **47**(9): p. 3754-65.

195. Weake, V.M., et al., *Post-transcription initiation function of the ubiquitous SAGA complex in tissue-specific gene activation.* Genes Dev, 2011. **25**(14): p. 1499-509.

196. Weake, V.M. and J.L. Workman, *SAGA function in tissue-specific gene expression.* Trends Cell Biol, 2011.

197. Hiller, M., et al., *Testis-specific TAF homologs collaborate to control a tissue-specific transcription program.* Development, 2004. **131**(21): p. 5297-308.

198. Matzat, L.H., et al., *Tissue-specific regulation of chromatin insulator function.* PLoS Genet, 2012. **8**(11): p. e1003069.

199. Ayroles, J.F., et al., *Systems genetics of complex traits in Drosophila melanogaster.* Nat Genet, 2009. **41**(3): p. 299-307.

200. Jumbo-Lucioni, P., et al., *Systems genetics analysis of body weight and energy metabolism traits in Drosophila melanogaster.* BMC Genomics, 2010. **11**: p. 297.

201. Geiger, T., et al., *Comparative proteomic analysis of eleven common cell lines reveals ubiquitous but varying expression of most proteins.* Mol Cell Proteomics, 2012. **11**(3): p. M111 014050.

202. Keshava Prasad, T.S., et al., *Human Protein Reference Database--2009 update.* Nucleic Acids Res, 2009. **37**(Database issue): p. D767-72.

203. D'Eustachio, P., *Reactome knowledgebase of human biological pathways and processes.* Methods Mol Biol, 2011. **694**: p. 49-61.

204. Celniker, S.E., et al., *Unlocking the secrets of the genome.* Nature, 2009. **459**(7249): p. 927-30.

205. Kerrien, S., et al., *Broadening the horizon--level 2.5 of the HUPO-PSI format for molecular interactions.* BMC Biol, 2007. **5**: p. 44.

206. Orchard, S., et al., *The minimum information required for reporting a molecular interaction experiment (MIMIx).* Nat Biotechnol, 2007. **25**(8): p. 894-8.

207. Aranda, B., et al., *PSICQUIC and PSISCORE: accessing and scoring molecular interactions.* Nat Methods, 2011. **8**(7): p. 528-9.

208. Schaefer, M.H., et al., *HIPPIE: Integrating protein interaction networks with experiment based quality scores.* PLoS One, 2012. **7**(2): p. e31826.

209. Neph, S., et al., *An expansive human regulatory lexicon encoded in transcription factor footprints.* Nature, 2012. **489**(7414): p. 83-90.

210. Vogel, C. and E.M. Marcotte, *Insights into the regulation of protein abundance from proteomic and transcriptomic analyses.* Nat Rev Genet, 2012. **13**(4): p. 227-32.

211. Ning, K., D. Fermin, and A.I. Nesvizhskii, *Comparative analysis of different label-free mass spectrometry based protein abundance estimates and their correlation with RNA-Seq gene expression data.* J Proteome Res, 2012. **11**(4): p. 2261-71.

212. Hornshoj, H., et al., *Transcriptomic and proteomic profiling of two porcine tissues using high-throughput technologies.* BMC Genomics, 2009. **10**: p. 30.

ABSTRACT

THE DROSOPHILA INTERACTIONS DATABASE: INTEGRATING THE INTERACTOME AND TRANSCRIPTOME

by

THILAKAM MURALI

August 2013

Advisor: Dr. Russell L. Finley Jr.

Major: Molecular Biology and Genetics

Degree: Doctor of Philosophy

In this thesis I describe the integration of heterogeneous interaction data for *Drosophila* into DroID, the *Drosophila* interactions database, making it a one-stop public resource for interaction data. I have also made it possible to filter the interaction data using gene expression data to generate context-relevant networks making DroID a one-of-a kind resource for biologists. In the two years since the upgraded DroID has been available, several studies have used the heterogeneous interaction data in DroID to advance our understanding of *Drosophila* biology thus validating the need for such a resource for biologists. In addition to this, I have identified organizing principles of interaction networks based on genome-wide gene expression data in the tissues and the entire life cycle of *Drosophila*. I have shown that all tissues and stages have a core ubiquitously expressed PPI network to which tissue and stage specific proteins attach to potentially modulate specific functions. In view of these organizing principles, I developed a normalized expression filter for interaction networks. I have shown that networks generated by using this filter are context-relevant as evidenced by their

enrichment for genes with relevant mutant phenotypes. This filter has been implemented in DroID and I anticipate that studies on interactome networks using this filter will further our understanding of biology.

AUTOBIOGRAPHICAL STATEMENT

THILAKAM MURALI

EDUCATION:

2007-2013 Ph.D in molecular biology and genetics, Wayne State University, Detroit, MI, USA

1993-1995 M.Sc in Biotechnology, Madurai Kamaraj University, Madurai, India

1990-1993 B. Sc in Biochemistry, Bharathiar University, Coimbatore, India

PUBLICATIONS:

Murali, T., Pacifico, S., Yu, J., Guest, S., Roberts, G. G. 3rd and Finley, R. L., Jr. DroID 2011: a comprehensive, integrated resource for protein, transcription factor, RNA and gene interactions for Drosophila, *Nucleic Acids Research. 2011;39(Database issue)*:D736-43.

Friedman, A. A., Tucker, G., Yan, D., Arunachalam, V., Hu, Y., Singh, R., Binari, R., Hong, P., Sun, X., Porto, M., Pacifico, S., **Murali, T.**, Finley, R. L., Jr., Asara, J. M., Berger, B., and Perrimon, N. Proteomic and functional genomic landscape of receptor tyrosine kinase and ras to extracellular signal-regulated kinase signaling., *Science Signaling 2011;4(196):rs10.*

Yu, J., **Murali, T.,** and Finley, R. L., Jr. Assigning Confidence Scores to Protein–Protein Interactions, *Methods in Molecular Biology 2012;812:161-74*.

Mairiang, D., Zhang, H., Sodja, A., **Murali T.**, Suriyaphol P., Malasit P., Limjindaporn T., Finley, R. L., Jr. (2013) Identification of new protein interactions between dengue fever virus and its hosts, human and mosquito, PLoS ONE 8(1): e53535. doi:10.1371/journal.pone.0053535

Murali, T., Pacifico S. and Finley, R. L., Jr. Integrating the interactome and transcriptome of Drosophila, *Manuscript in preparation*